Il BASIC in 30 ore per ZX81

Clive Prigmore

GRUPPO EDITORIALE JACKSON

Il BASIC in 30 ore per ZX81

Clive Prigmore

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

INDICE

Come usare questo corso

Scopo di questo corso
Per utilizzare con facilità e successo un microcomputer occorre conoscere tre cose:
(a) il linguaggio BASIC utilizzato dalla macchina;
(b) come realizzare delle buone strutture di programma;
(c) la tastiera. Questo corso vi insegna principalmente le prime due. Il vostro computer vi insegnerà la terza!
Il BASIC in 30 Ore non vi fa conoscere tutto sul BASIC, ma espone l'essenziale mettendovi in condizione, alla fine del corso di realizzare autonomamente dei programmi e, eventualmente, di perfezionare succesivamente la conoscenza del BASIC leggendo altri manuali.

Occorre avere il microcomputer?

È possibile seguire questo corso anche non disponendo di un microcomputer, comunque il migliore uso si ha se si lavora con uno ZX 81, poiché il contenuto del corso fa riferimento in modo particolare a questa macchina.
Il corso può essere svolto nel modo seguente:

Con uno ZX 81: fate tutti gli Esercizi e le domande di autovalutazione (TEST) ed inserite sul computer i programmi marcati con K .
Senza un microcomputer: fate tutti gli Esercizi ed i TEST e saltate tutto ciò che è marcato con K .

Struttura del corso

Il corso è costituito da 8 Unità, ed ogni Unità comprende:

Esempi: Sono problemi che nel testo vengono già risolti per voi.

Domande di autovalutazione (TEST): Vi chiederemo di fermarvi per controllare se avete compreso una nuova idea appena esposta. Le risposte vengono sempre date alla fine dell'Unità nella quale si trovano i TEST.

Esercizi: Questi sono dei problemi più lunghi che vi si chiede di svolgere la soluzione dei quali viene data sempre alla fine dell'Unità.

[K] sta per 'Key'. Questo simbolo è presente nei punti del testo nei quali può essere di aiuto inserire un programma nel vostro ZX 81.

CAPITOLO 1

SEMPLICI ISTRUZIONI E COMANDI

1.1 Cosa fa un computer?

In poche parole, il computer è una macchina che ci aiuta a risolvere certi tipi di problemi. Questi implicano usualmente simboli e caratteri che ci sono familiari attraverso il loro uso giornaliero, come le lettere dell'alfabeto (maiuscole e minuscole), i numeri, i simboli di interpunzione ed alcuni caratteri speciali come +, ,*, ecc. Il computer ci permette di inserire una serie di caratteri e ci ritorna una differente ma correlata di caratteri. Se ciò può apparire molto vago e generale, considerate alcuni esempi specifici.

CARATTERI INSERITI	CARATTERI RITORNATI
Numeri che rappresentano le dimensioni di una finestra.	Costo di una finestratura.
Lista dei libri presi in prestito in una libreria.	Lista dei libri da restituire.
Il nome di una persona.	Il numero di telefono della persona stessa.
Notazione standard di una mossa in una partita a scacchi.	Immagine di una scacchiera con la mossa eseguita.
Numeri che rappresentano pesi ed accelerazioni.	Immagine di un atterraggio lunare.
Codici predefiniti.	Pezzo musicale.

Figura 1 Alcuni usi di un computer

Questo corso non descrive come il computer fa queste cose ma in che modo voi potete ottenerle dandogli le giuste istruzioni. Non entreremo quindi in nessun dettaglio sull'interno di un computer, ma pensiamo che possa esservi di aiuto sapere quali sono le parti principali di un computer e ciò viene descritto in maniera semplice nel prossimo paragrafo.

1.2 Cos'è un computer?

Un semplice modello di computer e visibile in Figura 2.

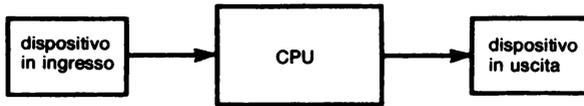



Figura 2 Un semplice modello di computer

Come potete notare, in un computer ci sono tre parti principali:

1 Il dispositivo di ingresso che vi consente di inserire sia le istruzioni che i dati (o informazioni) nel computer. In un microcomputer il dispositivo di ingresso è una tastiera simile a quella di una macchina da scrivere.

2 La central processing unit (CPU) o unità di elaborazione centrale, che, fra le altre cose, ritorna le istruzioni che avete inserito. Questo processo risulta dalla modificazione dei vostri dati dandovi la 'risposta' o il risultato che cercate.

3 Un dispositivo di uscita che vi consente di ricevere il risultato dell'elaborazione. Il dispositivo di uscita può essere lo schermo di un televisore che visualizza dei dati in uscita o una stampante che che stampa gli stessi dati su carta.

Tutto ciò può sembrare molto generico. In effetti non vengono indicate le tre caratteristiche chiave di un computer: (a) la sua capacità di memorizzare grandi quantità di dati che (b) egli è in grado di elaborare molto rapidamente e (c) la sua capacità di memorizzare un programma che controlla il suo stesso funzionamento. Quest'ultima caratteristica è la più importante ed è quella che descriviamo in questo corso.

Memoria su cassette

Prima di passare alla programmazione, vorremmo menzionare un'altra sola caratteristica tecnica. Se state usando questo corso disponete probabilmente di un microcomputer con una piccola capacità di memorizzazione interna. Il computer usa questa memoria per tenere le istruzioni ed i dati del problema che stà risolvendo in quel momento. Tali istruzioni vengono però cancellate quando la macchina

viene spenta, per cui se desiderate conservare i vostri programmi o dati, dovete memorizzarli su una memoria esterna generalmente costituita da una cassetta. Il registratore per scrivere o leggere tali cassette può essere collegato al computer, quando necessario, tramite degli appositi cavi. Nei sistemi di grandi dimensioni inoltre, i programmi possono essere conservati su dischi magnetici invece che su cassette.

Per riassumere, i principali elementi di un computer sono illustrati in Figura 3.

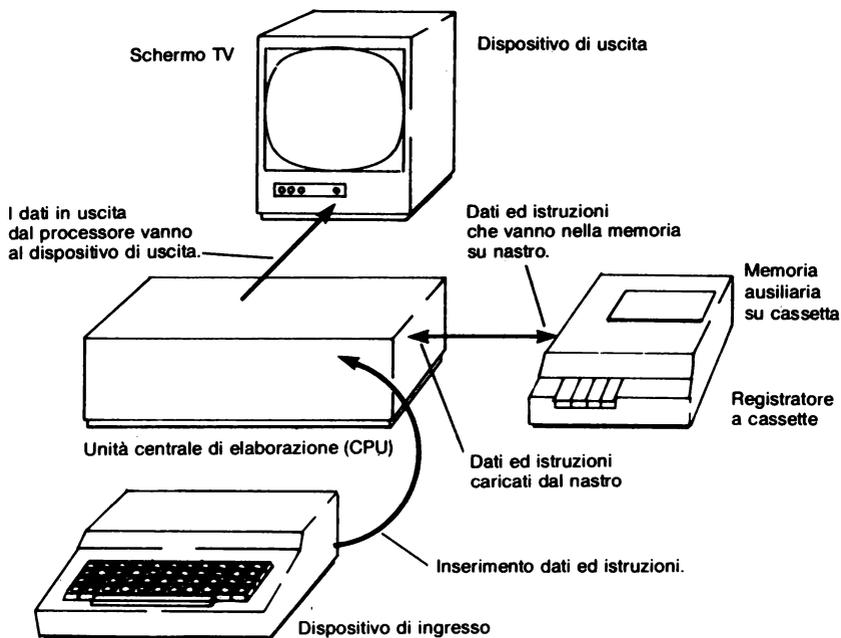



Figura 3 Un tipico sistema di computer

1.3 Cos'è il BASIC

Il computer è un dispositivo elettronico che elabora delle configurazioni di segnali elettrici. Se avete un problema non potete certo inserirlo nel computer con dei segnali elettrici, ne il computer vi fornisce il risultato dell'elaborazione sotto forma di segnali elettrici, questo perché il computer ha un codice macchina al suo interno (inserito dal produttore) che gli consente di capire un codice di programmazione

che anche voi potete facilmente comprendere. I codici macchina sono chiamati linguaggi di programmazione a basso livello e corrispondono direttamente con le configurazioni dei segnali elettrici. Per ovvie ragioni, questo programma è chiamato interprete. Questo corso vi insegna il BASIC che è un linguaggio ad alto livello. Voi sarete quindi in grado di usare il BASIC per programmare qualsiasi computer che contiene un interprete BASIC. BASIC stà per Beginners' All-purpose Symbolic Instruction Code (Codice Simbolico di Istruzioni per tutti gli usi per principianti).

Può essere utile notare la sequenza di eventi che avvengono quando programmate un computer.

1 Avete un problema.
2 Suddividete il problema in una serie di passi eseguibili con il BASIC.
3 Scrivete il programma in BASIC.
4 Vi sedete di fronte alla tastiera ed inserite il programma nel computer.
5 Il computer interpreta le vostre istruzioni in BASIC nel proprio codice e le elabora.
6 Il computer stampa il risultato nella forma da voi specificata nel programma.

Questo è tutto quello che dovete conoscere su cosa è un computer. Da ora in poi noi terremo presente che tutto ciò che dovete fare è dare al computer un problema e ricevere da lui un risultato, quindi ora passiamo a descrivere un semplice problema che noi potremmo dare ad un computer.

1.4 Un semplice problema

La principale attività nella programmazione e quella di suddividere il problema in passi più semplici che possono essere rappresentati da istruzioni BASIC.

Immaginate di interpretare la parte di un computer con un ragazzo. Il ragazzo deve darvi due numeri e chiedervi di dargli la somma. Dopo un po di tempo il ragazzo potrebbe darvi dei numeri molto grandi che non sarete più in grado di sommare nella vostra testa, per cui avete bisogno anche di un foglio di carta e di una penna. Il seguente può essere un tipico dialogo tra voi (che fate il computer) e il ragazzo:

RAGAZZO: 'Parti'
VOI: 'Dammi il primo numero'
RAGAZZO: '12157'
(Scrivete il primo numero su un pezzo di carta)
VOI: 'Dammi il secondo numero'

RAGAZZO: '7896'
(Scrivete anche il secondo numero sul pezzo di carta)
(Sommate i due numeri)
VOI: '20053'

Possiamo descrivere la parte del computer in questo processo più formalmente nel seguente modo:

1 Inserire il primo numero
2 Inserire il secondo numero
3 Sommare i due numeri
4 Fornire il risultato

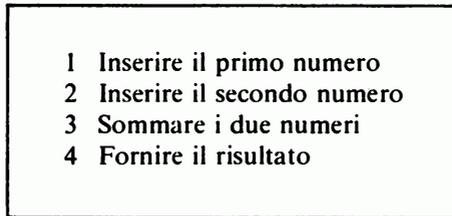



Figura 4 Sequenza del computer nella somma di due numeri

Da questa semplice analogia siamo arrivati ad una strategia per risolvere questo problema. In altre parole, le fasi 1 e 2 concernono l'inserimento di numeri nel computer, la fase 3 coinvolge nel processo l'unità di elaborazione centrale (CPU) e la fase 4 coinvolge il dispositivo di uscita.

Ora, anche se non abbiamo ancora descritto la programmazione in BASIC, mostreremo come appare la sequenza per risolvere il problema dopo averla scritta in BASIC.

Esempio 1

Scriviamo un programma in BASIC per inserire due numeri nel computer e ritornare la loro somma.

Soluzione

Abbiamo già fornito una procedura intuitiva per risolvere questo problema in Figura 4. Un programma in BASIC può avere la seguente forma:

```
10 INPUT PRIMO
20 INPUT SECONDO
30 LET SOMMA=PRIMO+SECONDO
40 PRINT SOMMA
50 STOP
```

Programma 1

Non vogliamo concentrarci in questo momento sui dettagli del Programma 1, ma speriamo che abbiate notato come la strategia di Figura 4 è stata trasformata in un programma. Un programma è, quindi una sequenza di istruzioni creata per risolvere un dato problema con il computer.

TEST 1

Ora per la prima volta in questo corso vogliamo verificare i vostri progressi attraverso delle Domande di Autovalutazione (TEST). I TEST sono stati scritti per aiutarvi a sapere cosa avete o non avete imparato dalla sezione del corso appena letta. In ogni caso le risposte ai TEST vengono date alla fine dell'Unità nella quale si trovano. Se date tutte risposte esatte passate alla prossima sezione. Se invece avete fatto un errore tornate indietro per scoprire dove avete sbagliato.

Scegliete quelle frasi dalla lista B che completano correttamente le frasi date in A.

A
1 La CPU ...
2 Le principali caratteristiche di un computer sono ...
3 Un codice macchina è ...
4 Un codice macchina è un esempio di linguaggio a ...
5 Il BASIC è un esempio di linguaggio ad ...
6 Un interprete BASIC ...
7 Un programma è ...

B
(a)... basso livello.
(b)... alto livello.
(c)... ritiene dati ed istruzioni, controlla il proprio funzionamento e controlla il funzionamento dei dispositivi di ingresso ed uscita.
(d)... una serie di istruzioni che rappresentano i passi da compiere per risolvere un dato problema.
(e)... la capacità di memorizzare grandi quantità di dati, la capacità di elaborare tali dati molto rapidamente ed inoltre la possibilità di memorizzare programmi che controllano il suo funzionamento.
(f) ... traduce i codici scritti in BASIC nei codici macchina.
(g)... un codice che corrisponde direttamente con la forma dei segnali elettrici all'interno del computer.

1.5 Numeri delle istruzioni

Date uno sguardo più accurato al Programma 1:

```
10 INPUT PRIMO
20 INPUT SECONDO
30 LET SOMMA=PRIMO+SECONDO
40 PRINT SOMMA
50 STOP
```

Programma 1 (da pag. 11)

Noi abbiamo detto che un programma è una sequenza di istruzioni. Nel programma qui sopra ogni linea è un'istruzione. Quindi:

```
10 INPUT PRIMO
```

è la prima istruzione del programma, e

```
50 STOP
```

l'ultima.

Inserimento delle istruzioni

Il processo che permette di inserire delle istruzioni ha inizio dopo che viene premuto il tasto NEW LINE. La sequenza è:
Scrivete per primo 10 INPUT e premete NEW LINE. Quindi scrivete 20 INPUT per secondo e premete NEW LINE ecc.

Ora voi avete sullo schermo:

```
10 INPUT PRIMO
20 INPUT SECONDO
```

Come potete notare ogni riga inizia con un numero. Questo deve essere un numero intero compreso tra 1 e 9999, ed esso determina l'ordine nel quale le istruzioni vengono elaborate (eseguite), in altre parole definisce la 'sequenza' delle istruzioni. L'esecuzione delle istruzioni parte con la riga di numero più basso e continua con le righe di numero sempre maggiore fino a che non venga richiesto diversamente e fino a che non si arriva alla fine del programma (presto descriveremo anche questi casi).

Allora perché, potrete domandare, il programma non è stato scritto nel modo seguente?:

```
1 INPUT PRIMO
2 INPUT SECONDO
3 LET SOMMA=PRIMO+SECONDO
4 PRINT SOMMA
5 STOP
```

Programma 2

Certamente! Il programma avrebbe fatto lo stesso lavoro perfettamente! Comunque, come potrete notare quando scrivete i programmi, è meglio che abbiate una certa flessibilità. In particolare per avere la possibilità di inserire nel programma una riga che avete dimenticato o per consentirvi di effettuare delle modifiche importanti al programma. Numerando le nostre righe 10, 20, 30, 40 e 50 possiamo disporre, tra una istruzione e l'altra, di 9 righe vuote che possono essere usate per correggere o modificare il programma.
Durante il funzionamento, l'elaborazione procede sempre verso la riga successiva di numero più elevato, così che il fatto che vi siano nove numeri di riga inutilizzati non rallenta l'esecuzione del programma in alcun modo.

TEST 2

Osservate i numeri di riga dei seguenti programmi e decidete quali programmi possono produrre la somma di 'primo' e 'secondo' correttamente.

(a)
```
 11 INPUT PRIMO
 59 INPUT SECONDO
 33 LET SOMMA=PRIMO+SECONDO
401 PRINT SOMMA
500 STOP
```

(b)
```
23 INPUT PRIMO
32 INPUT SECONDO
40 PRINT SOMMA
49 LET SOMMA=PRIMO+SECONDO
50 STOP
```

(c)

```
10 INPUT PRIMO
15 LET SOMMA=PRIMO+SECONDO
20 INPUT SECONDO
40 PRINT SOMMA
50 STOP
```

(d)

```
100 INPUT PRIMO
110 LET SOMMA=PRIMO+SECONDO
190 PRINT SOMMA
200 INPUT SECONDO
220 STOP
```

(e)

```
  50 INPUT SECONDO
 100 INPUT PRIMO
 407 LET SOMMA=PRIMO+SECONDO
 902 PRINT SOMMA
1000 STOP
```

Programmi 3-8

1.6 Esecuzione e comandi

Il comando RUN

L'esecuzione? No, non è la fine ma l'inizio quindi facciamo girare il nostro primo programma prima di annoiarci a vederlo!

```
10 INPUT PRIMO
20 INPUT SECONDO
30 LET SOMMA=PRIMO+SECONDO
40 PRINT SOMMA
50 STOP
```

Programma 1 (da pag 11)

E cosa succede? Niente. Questo perché il computer è in attesa di ricevere da noi delle istruzioni da dare al programma. Se volete esguire questo programma, dovete dare il comando di RUN.

```
10 INPUT PRIMO
20 INPUT SECONDO
30 LET SOMMA=PRIMO+SECONDO
40 PRINT SOMMA
50 STOP
RUN
```

Programma 1 (da pag 11)

(Non preoccupatevi del fatto che RUN non ha un numero di riga - spiegheremo presto questo fatto.)

Quindi premete NEW LINE. Voi vedrete allora una L lampeggiante sullo schermo che è il modo del computer di chiedere l'inserimento di un dato. Dategli il vostro primo numero e premete NEW LINE, vedrete apparire un'altra L lampeggiante perché il computer vi chiede di inserire il secondo numero. Dategli il secondo numero e premete NEW LINE. Ora dovrebbe apparire la risposta. Questa è la nostra versione di tale sequenza:

```
10 INPUT PRIMO
20 INPUT SECONDO
30 LET SOMMA=PRIMO+SECONDO
40 PRINT SOMMA
```

9/50 ______ significa STOP alla linea 50

Figura 5 Esecuzione completa di un programma

Quello che dobbiamo fare, perciò, e di distinguere tra l'inserimento di un programma e la sua esecuzione. Torniamo ora al dialogo tra voi ed il ragazzo che usa il computer. Egli può dirvi 'Ora sto per darti due numeri, voglio che tu te li scriva, che li sommi tra loro, e mi dai il risultato della loro somma'. A questo punto voi sapete esattamente cosa fare ma non avete ancora fatto niente; siete stato programmato. Il dialogo può continuare così:

RAGAZZO: 'Parti'
VOI: 'Dammi il primo numero'.
RAGAZZO: '12157'
VOI: 'Dammi il secondo numero'.
RAGAZZO: '7896'
VOI: '20053'

Adesso queste istruzioni sono state eseguite (run). Un programma è solo una serie di istruzioni per il computer. Quando il programma viene fatto girare queste istruzioni vengono eseguite.

Altri comandi: LIST, SAVE, LOAD

RUN non è l'unico comando che può essere dato al computer direttamente. Potete usare anche LIST, SAVE e LOAD nel seguente modo:

LIST

Potete impiegare un certo tempo per inserire un programma nel vostro computer e, durante questo processo, fare diverse correzioni, volete quindi vedere il programma per intero sullo schermo. Se inserite la parola LIST, sullo schermo apparirà una copia del programma ordinata secondo il numero di linea. (Con programmi più lunghi appare il messaggio 5/0 quando lo schermo è pieno. Questo significa 'fine dello spazio sullo schermo'. Per vedere la prossima sezione del programma sullo schermo, premete LIST seguito dall'ultimo numero di linea uscito sullo schermo ed il listato continuerà da questa linea.)

SAVE

Dopo aver sviluppato un programma fino ad un punto soddisfacente potreste volerne conservare una copia su nastro; la parola SAVE (seguita dal nome che volete dare al programma tra virgolette, per es. SAVE "DAMA") permette di far questo. (Il vostro computer dovrà, naturalmente, essere collegato ad un registratore a cassette.)

LOAD

In seguito potreste voler usare uno dei programmi che avete conservato su cassetta; il comando LOAD "DAMA" caricherà il programma dalla memoria ausiliaria su cassetta alla memoria del computer.

Parole come LIST, RUN, SAVE e LOAD, che ci consentono di trattare il programma come una singola entità, sono chiamati comandi e sono messi a disposizione dall'interprete BASIC. Un comando occupa una riga a sé e generalmente non ha un numero di riga, per es. la parola RUN dopo la riga 50 del Programma 1 causa l'inizio dell'esecuzione del programma, ed equivale al comando di 'Parti' dato dal ragazzo nel dialogo descritto precedentemente. I Comandi verranno discussi più dettagliatamente in una delle prossime Unità, ma i quattro comandi che abbiamo già visto ci consentono di essere pronti a partire.

Keywords

Queste sono le parole del BASIC che specificano nel programma quale azione deve essere eseguita, per es. INPUT, PRINT, LET.

1.7 Esecuzione e dati

Vi sarete resi conto che abbiamo scritto un programma che addiziona due numeri qualsiasi in modo del tutto generale, perciò è assolutamente indifferente quali numeri noi inseriamo quando riceviamo il segnale di 'pronto' L sullo schermo, il computer esegue il programma essendo in grado di richiedere l'inserimento di numeri reali per un determinato lavoro, vale a dire deve avere la possibilità di chiedere dei dati relativi al lavoro che deve svolgere. Voi dovete distinguere chiaramente tra il programma, che è una serie di istruzioni più o meno generali, e i dati che sono i numeri reali che devono essere inseriti quando il programma è in esecuzione per risolvere un particolare problema. Voi potete, naturalmente, far girare il vostro programma più volte con molte coppie di numeri, come vedrete più avanti.

Un'altro modo di vedere queste istruzioni è di visualizzare la situazione come un un arbitro che riunisce i corridori alla partenza di una corsa per dare loro delle istruzioni: 'Andate giù per il lato destro del campo fino al prossimo angolo, oltre la scaletta e dirigetevi a sinistra per il sentiero ....' Le istruzioni dell'arbitro sono simili ad un programma. Se i corridori capiscono quello che dice allora sanno cosa fare; ma sono fermi alla linea di partenza, non sono ancora partiti. Questo è analogo a quando il programma è stato inserito nella macchina. Quindi l'arbitro dice 'Via', la corsa inizia. Questo è analogo a quando il computer comincia ad eseguire il programma. Estendete l'analogia e considerate la corsa attraverso i campi come una corsa con delle novità. Immaginate che l'arbitro non dia abbastanza istruzioni ai corridori per completare la corsa ma che egli dica cose come 'Quando siete alla fine del sentiero troverete ulteriori istruzioni attaccate ad una quercia ....' Queste istruzioni dovrebbero essere sufficienti per guidare i corridori fino alla parte successiva della corsa, vale a dire fino alla prossima traccia, e così via fino alla fine della corsa. Queste tracce sono analoghe al dare al programma dei dati in più durante la sua esecuzione. Questa analogia può aiutarvi a fare una distinzione importante tra l'inserimento di un programma nella macchina, l'esecuzione del programma, e quindi l'inserimento di dati durante la sua esecuzione.

TEST 3

Qui sotto è visibile la stampa di un programma che contiene delle keywords, dei comandi, delle risposte del computer, e dei dati. Inoltre esso è costituito da sezioni che concernono con l'inserimento del programma, l'esecuzione ed il listing. Identificate il maggior numero possibile di queste cose nel modo seguente:

Segnate le keywords con K
Segnate i comandi con C
Segnate le risposte del computer con R
Segnate i dati con D
Mettete tra parentisi le righe che concernono l'inserimento del programma
Mettete tra parentesi le righe che concernono l'esecuzione
Mettete tra parentesi le righe che concernono il listing

```
10 INPUT PRIMO
20 INPUT SECONDO
30 LET SOMMA=PRIMO+SECONDO
40 PRINT SOMMA
50 STOP

RUN
L-37
L-46
-83

0/50
LIST

10 INPUT PRIMO
20 INPUT SECONDO
30 LET SOMMA=PRIMO+SECONDO
40 PRINT SOMMA
50 STOP

RUN
L12.83
L48.95
61.78

0/50

RUN
17.0009
29.3629
-12.362

0/50
```

Programma 1 (da pag 11)

1.8 INPUT, PRINT e LET

Avete visto che un programma è una seqenza di istruzioni, vi abbiamo anche dato un'idea intuitiva di come funziona ogni istruzione. Potete avere notato che ciascuno dei tre tipi di istruzioni appena usate (INPUT, LET e PRINT) corrisponde ad uno dei tre principali dispositivi compresi nel sistema computer (dispositivo di ingresso, elaboratore centrale e dispositivo di uscita). Ora daremo uno sguardo a ciascuna istruzione più in dettaglio.

INPUT

La parola INPUT è per il computer un segnale che durante l'esecuzione, un dato deve essere inserito tramite il dispositivo di ingresso. Abbiamo visto accadere ciò facendo girare il nostro primo programma: dopo la L abbiamo inserito 12157 ,premuto NEW LINE per completare la procedura e quindi ci siamo trovati di fronte ad un'altra L che ci chiedeva di inserire il numero seguente. Cosa è successo, allora a 12157 che è stato il primo dato inserito? È che esso è stato conservato in una locazione di memoria etichettata con 'PRIMO' per essere utilizzato più avanti durante l'esecuzione del programma. La parola 'PRIMO' ha due principali funzioni nel programma; (a) quando viene scritta e successivamente il programmatore vi fa riferimento, gli ricorda che a quel punto del programma il primo dato dovrebbe essere inserito, e (b) quando viene scritta nell'istruzione 10 INPUT PRIMO è il nome o l'etichetta di una locazione della memoria del computer. Così 10 INPUT PRIMO vuol dire 'ricevi un numero dal dispositivo di ingresso e memorizzalo in una locazione etichettata 'PRIMO''.

PRINT

L'istruzione 40 PRINT SOMMA ha praticamente l'effetto inverso delle istruzioni 10 e 20, in quanto ci consente di ricevere informazioni dalla macchina. È un segnale che chiede alla macchina di fare una copia del contenuto della locazione di memoria etichettata con 'SOMMA' e di passarla al dispositivo di uscita che per molti lettori di questo corso sarà lo schermo del televisore. Notate che solo LPRINT dà in realtà una copia stampata (PRINTed) se al computer è collegata una stampante ma voi usate tranquillamente il comando PRINT quando il computer è collegato al televisore come dispositivo di uscita.

LET

30 LET SOMMA=PRIMO+SECONDO è un esempio di una istruzione di assegnazione. È in questo tipo di istruzione che ha luogo l'elaborazione. Come potete vedere, è una mistura di nomi di locazioni ('SOMMA', 'PRIMO', 'SECONDO') e di operatori aritmetici (= e +). Se la leggete nel modo seguente, dice;

'Fai in modo (Let) che la locazione 'SOMMA' diventi uguale alla locazione 'PRIMO' addizionata alla locazione di nome 'SECONDO''.

Comunque, come molte espressioni matematiche, è normalmente più chiara se letta dalla destra alla sinistra del segno '=' così:

'Somma il contenuto della locazione 'PRIMO' al contenuto della locazione 'SECONDO' e conserva il risultato nella locazione 'SOMMA''.

Generalmente l'istruzione di assegnazione ha il formato:

LET nome locazione di memoria = espressione

Questo vuol dire calcola il valore dell'espressione alla destra di '=' e conserva questo valore nella locazione di memoria indicata alla sinistra di '='.

Il punto delicato a proposito di LET ...=... è che può essere facilmente confusa con ...=... nelle equazioni matematiche. Un esempio può chiarire la differenza. Supponiamo che abbiate memorizzato un numero nella locazione L e che vogliate far diventare tale numero più grande di 5 rispetto ad adesso. Scrivete:

LET L=L+5

Ora ovviamente questo non significa:

L=L+5

poiché non c'è alcun valore di L che possa far ciò veramente. Quello che significa è invece che il computer ha sommato 5 al numero che è contenuto nella locazione L .

1.9 Locazioni di memoria

Come abbiamo già detto, una delle principali caratteristiche di un computer è la capacità di conservare grandi quantità di dati. Dobbiamo ora considerare in che modo il linguaggio BASIC ci consente di assegnare i nomi delle locazioni di memoria. Se date uno sguardo al nostro primo programma e ricordate che un computer è capace di fare solo una cosa alla volta, è evidente che quando noi arriviamo alla riga 20 e vogliamo inserire il nostro secondo numero, il numero che abbiamo inserito con l'istruzione della riga 10 deve essere stato conservato da qualche parte in questo caso il primo numero è stato conservato nella locazione di nome 'PRIMO'. Noi possiamo pensare alle locazioni di memoria come se si

trattasse di una serie di scatole dove possiamo distinguere chiaramente tra l'etichetta o l'indirizzo o il nome di ciascuna scatola ed il suo contenuto.

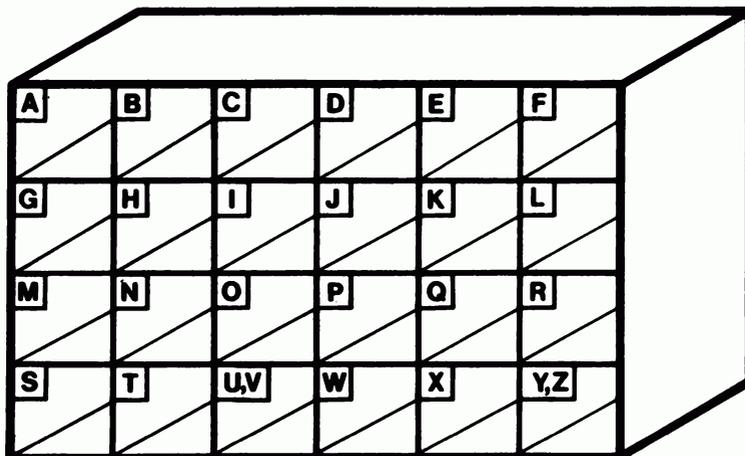



Figura 6 Modello delle locazioni di memoria in un piccolo computer

Come potete vedere, nel nostro modello delle possibili locazioni di memoria, abbiamo usato le etichette A, B, C ... D'altra parte, nel programma che abbiamo studiato sono state usate parole lunghe fino a sette caratteri per etichettare le locazioni, per es. 'PRIMO', 'SECONDO'. Questo ci porta ad una delle principali differenze tra gli interpreti dei vari microcomputers oggi disponibili. Alcuni interpreti BASIC consentono una maggiore scelta di nomi di altri. Alcune macchine limitano i nomi delle vostre variabili a pochi caratteri. Altre consentono nomi delle variabili più lunghi o ancora nomi senza restrizioni nella lunghezza. Lo ZX 81 consente nomi di variabili di lunghezza illimitata.

Come scegliere i nomi delle locazioni

Chiaramente è molto più comodo per il programmatore scegliere per le locazioni di memoria dei nomi che gli ricordino che cosa contengono. Per questo abbiamo scelto i nomi PRIMO, SECONDO e SOMMA. Avremmo potuto usare anche A, B ed S in questo caso il nostro programma sarebbe stato:

```
10 INPUT A
20 INPUT B
30 LET S=A+B
40 PRINT S
50 STOP
```

Programma 9

K Quando vedete questo simbolo vuol dire che vi suggeriamo di provare questo programma sul vostro microcomputer se ne avete uno a disposizione. Per far ciò, inserite le linee, premendo NEW LINE dopo ognuna, e per finire premete RUN e quindi ancora NEW LINE. Il vostro computer vi chiederà un numero. Dategliene uno e premete NEW LINE. Quindi vi chiederà il secondo numero. Dategli anche il secondo numero, premete NEW LINE e la somma apparirà sullo schermo. Se abbiamo fatto ciò, dopo avere inserito 12157 e 7896, le locazioni di memoria dovrebbero essere:

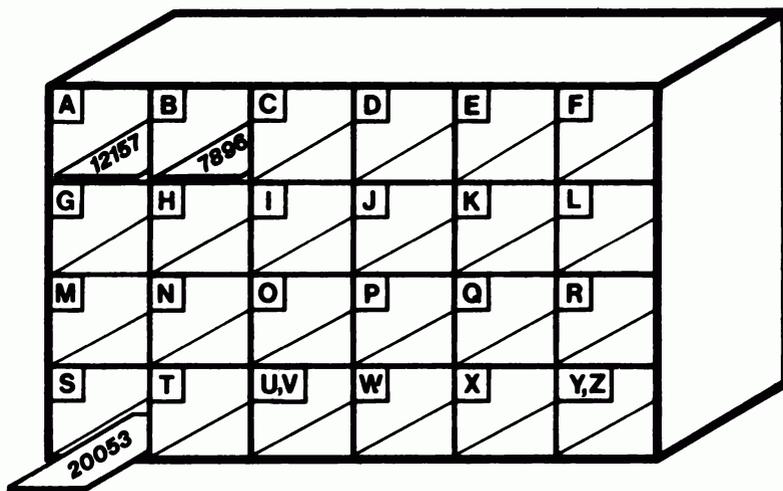



Figura 7 Stato delle locazioni di memoria dopo il programma 9

Sistema utilizzato per i nomi delle locazioni

Poiché i nomi lunghi per le locazioni non sono possibili su molti microcomputer, noi useremo, da ora in poi, per le locazioni di memoria dei nomi che vanno bene praticamente per qualsiasi microcomputer. Il sistema che useremo etichetta una locazione con una lettera maiuscola seguita da un numero. Questo ci dà 286 possibili locazioni, come visibile in Figura 8.

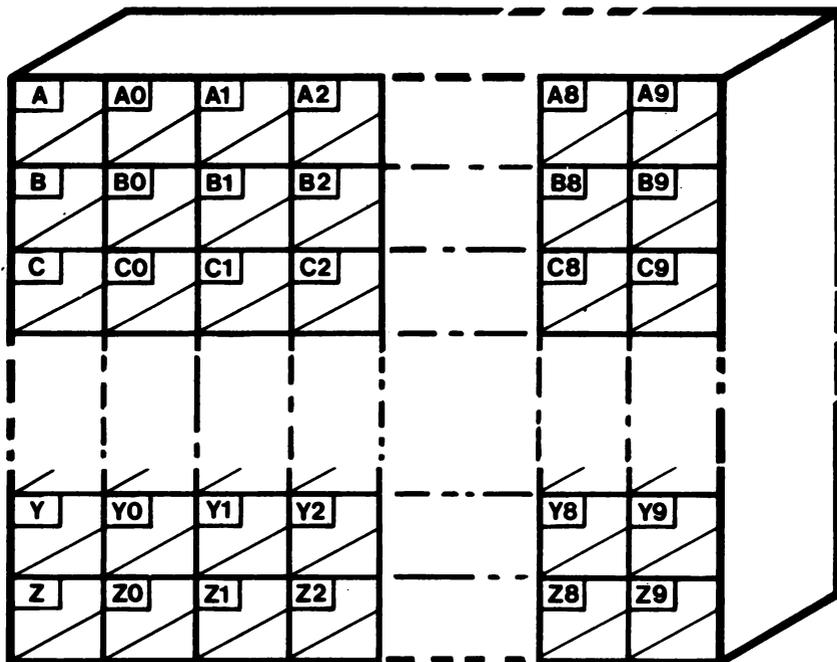
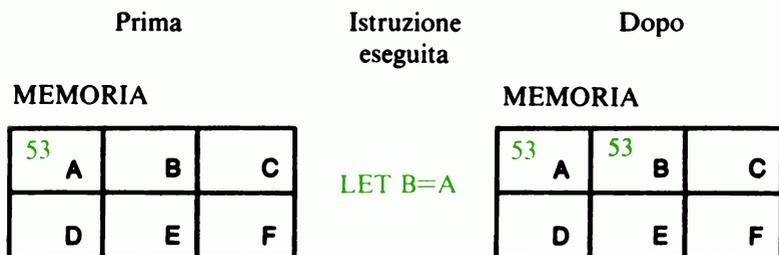



Figura 8 Le 286 possibili locazioni di memoria

1.10 Copiatura e sovrascrittura

Una istruzione BASIC può avere due diversi effetti sul contenuto di una locazione di memoria. O più esattamente, una istruzione può non avere nessun effetto sul contenuto della locazione o cambiare il contenuto stesso. Ciò viene illustrato di seguito.

Effetto della copiatura

Supponete che abbiamo il numero 53 memorizzato nella locazione A. Cosa accade dopo LET B=A e dopo PRINT A? In entrambe i casi A conserva il numero 53 dopo che le istruzioni sono state eseguite:

In ciascun caso l'istruzione di copiatura lascia inalterata la locazione di memoria originale. È semplicemente come chiedere l'ammontare del vostro conto in banca: il pezzo di carta copia la cifra del vostro conto ma il denaro che si trova sul vostro conto rimane!

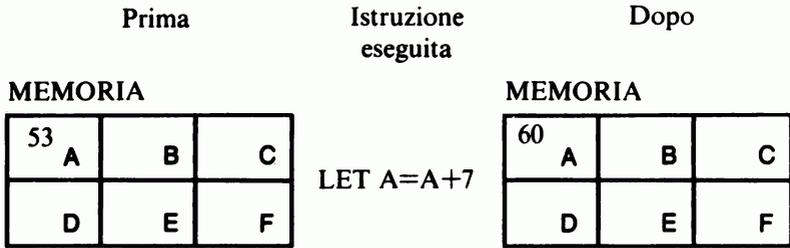
Effetto della sovrascrittura

Supponete di avere sempre il numero 53 memorizzato nella locazione A ma che questa volta venga eseguita l'istruzione LET A=A+7. Il risultato è:

Prima	Istruzione eseguita	Dopo
MEMORIA		MEMORIA

53 A	B	C
D	E	F

LET A=A+7

60 A	B	C
D	E	F

L'istruzione LET A = A + 7 sovrascrive il contenuto di A. Che vuol dire che il contentenuto originale sparisce e viene sostituito da un nuovo contenuto, che in questo caso è 60. Naturalmente noi avremmo potuto cambiare il contenuto di A in 60 in diversi modi, per es. con LET A=60.

TEST 4

Quali dei seguenti nomi di locazioni di memoria sono validi in base alle regole descritte nelle pagine precedenti?

(a) N3
(b) 3N
(c) W10
(d) B
(e) QJ
(f) M
(g) M5
(h) M-5
(i) M+5
(j) U0

Date anche la ragione per cui scartate alcuni dei nomi

1.11 Operatori aritmetici

Quando impiegate l'aritmetica utilizzate quattro principali operatori: +, −, × e ÷.
Il BASIC ha gli stessi operatori, sebbene due siano stampati in maniera diversa:

Simboli di ogni giorno	Significato	Simboli BASIC
+	addizione	+
−	sottrazione	—
×	moltiplicazione	*
÷	divisione	/

TEST 5

Scrivete le seguenti espressioni utilizzando gli operatori aritmetici del BASIC. (Dove le espressioni usano le parentesi lasciatele anche nelle vostre risposte.)

(a) 3+7
(b) 3×7
(c) 8÷4
(d) 5×(2+8)
(e)30÷(3+2)
(f)24−(4×3)
(g)5×6×7
(h)81-(27×2)

TEST 6

Se A ha il valore di 2, B ha il valore di 5 e C ha il valore di 10, calcolate il valore delle seguenti equazioni:

(a) A+B+C
(b) A*B
(c) A*B*C
(d) C/A
(e) C/(B-A)
(f) A*A
(g) (B*C)/(B-A)
(h) (C-B)*(C+B)

Le operazioni sono eseguite nel BASIC attraverso le istruzioni di assegnazione (istruzione LET) che dicono all'unità aritmetica del computer (parte dell'elaboratore centrale) che cosa deve fare. Noi possiamo illustrare ciò tramite il seguente modello di computer.

Effetto di LET A=B-C

MEMORIA ALL'INIZIO

A	15 B	10 C
D	E	F

LET A=B-C

Ricordate di leggere questa istruzione dalla destra alla sinistra del segno '='. Che vuol dire prendi il numero nella locazione B, sottrai da esso il numero nella locazione C e metti il risultato nella locazione A. Così il risultato è:

Risultato di LET A=B-C

MEMORIA ALLA FINE

5 A	15 B	10 C
D	E	F

Notate che i contenuti di B e C non sono cambiati.

Effetto di LET A=B*C

MEMORIA ALL'INIZIO

A	15 B	10 C
D	E	F

MEMORIA ALLA FINE

150 A	15 B	10 C
D	E	F

TEST 7

Riempite con i valori giusti le locazioni di memoria A, B e C dopo che è stata eseguita ciscuna linea dei seguenti programmi.

1. Programma — Valori locazioni di memoria

Programma	A	B	C
10 LET A=12	□	□	□
20 LET B=5	□	□	□
30 LET C=A*(A+B)	□	□	□
40 LET A=A+10	□	□	□

2. Programma — Valori locazioni di memoria

Programma	A	B	C
10 LET A=20	□	□	□
20 LET B=A*3	□	□	□
30 LET C=A/4	□	□	□
40 LET A=B+C	□	□	□

Quello che abbiamo appena fatto non è (speriamo) difficile e non aggiungiamo alcuna difficoltà in più se passiamo a nomi di locazioni più complicati. Noi dobbiamo usare dei nomi più complicati perché A, B, C, ecc... ci danno solo 26 locazioni diverse, così come abbiamo detto in precedenza, ora useremo nomi di locazioni tipo A, A0, A1, ecc. Così,

LET P4 = Q1*R1

non è diversa da LET A=B*C . P4, Q1 e R1 sono semplici nomi di locazioni. P4 è un nome così come MI 79832N è il numero di un'auto.

Noi siamo ora pronti per utilizzare le capacità aritmetiche di un computer.

Esempio 2

Scriviamo un programma BASIC per inserire due numeri nel computer e quindi ottenere la loro somma, differenza, prodotto e quoziente.

Soluzione

Questo problema può apparire complicato ma noi in realtà lo abbiamo già risolto. Noi abbiamo un programma (Programma 9) per ottenere la somma di due numeri, per cui per ottenere la loro differenza, prodotto e quoziente, dobbiamo cambiare soltanto l'operatore aritmetico nella riga 30 che si legge

```
30 LET SOMMA=PRIMO+SECONDO
```

Per prima cosa, riscriviamo il programma utilizzando dei nomi più corti per le locazioni:

Versione originale

```
10 INPUT N1
20 INPUT N2
30 LET S=N1+N2
40 PRINT S
50 STOP
```

Nuova versione

```
10 INPUT N1
20 INPUT N2
30 LET D=N1-N2
40 PRINT D
50 STOP
```

Programma 10

Ora ciò di cui abbiamo bisogno è di altre tre copie della nuova versione del programma, ciascuna con una differente riga 30:

10 INPUT N1 20 INPUT N2 30 LET S=N1+N2 40 PRINT S	" " " " " D=N1-N2 " D	" " " " " P=N1*N2 " P	" " " " " Q=N1/N2 " Q

(Usando D per la locazione della differenza, P per la locazione del prodotto e Q per la locazione del quoziente.)

Per far questo abbiamo bisogno di quattro programmi? Fortunatamente no, perché quando copiamo i numeri dalle locazioni N1 ed N2 non distruggiamo il loro contenuto così che noi possiamo usarli quattro volte o anche più all'interno di un programma:

Programma 11 Somma, differenza, prodotto e quoziente di due numeri

K Inserite il Programma 11 nel vostro microcomputer. Quindi date il RUN ed inserite i due numeri. Un tipico risultato potrebbe apparire in questo modo:

RUN

1.12 Costanti numeriche

All'inizio di questa Unità abbiamo visto che il nostro primo programma BASIC era capace di manipolare numeri interi, decimali e negativi. A questo stadio noi non entreremo in dettaglio su come vengono rappresentati i numeri in BASIC, ma vi faremo vedere solo come possiamo utilizzare dei numeri direttamente nelle istruzioni di assegnazione.

L'istruzione LET P=427*R vuol dire creare il numero 427, moltiplicarlo per il numero trovato nella locazione R e quindi memorizzare il risultato nella locazione P.

(Non fatevi sviare pensando che i computers possono manipolare solo numeri binari. L'interprete del computer ci consente di inserire dei normali numeri decimali.)

Similmente, l'istruzione LET Y4=3.142+Z8 crea il numero 3.142 lo addiziona al contenuto della locazione Z8 e quindi memorizza la somma nella locazione Y4.

E l'istruzione LET A=-48.93/B crea il numero -48.93 lo divide per il numero che si trova nella locazione B e quindi memorizza il risultato di questo calcolo nella locazione A.

Premessa sugli Esercizi

Nei paesi anglosassoni nei quali si sta cercando di introdurre il sistema metrico decimale, molte persone incontrano delle difficoltà nel convertire quotidianamente i chilogrammi nelle più familiari libbre, o ancora le yarde in metri, le pinte in litri, le once in grammi, i gradi Fahrenheit in gradi Centigradi o Celsius, ecc.. I prossimi due esercizi riguardano appunto la scrittura di programmi che fanno delle conversioni. Per svolgere tali esercizi occorre soltanto che teniate presenti le idee già introdotte nella parte iniziale di questa Unità.

Esercizio 1

Scrivete dei programmi BASIC che producano le seguenti conversioni:

(a) Inserito un numero che rappresenta una lunghezza in pollici il computer deve dare come risultato la stessa lunghezza in centimetri, dato che un pollice corrisponde a 2.54 centimetri.

(b) Inserito un numero che rappresenta un peso in once il computer deve ritornare lo stesso peso in grammi, dato che un oncia corrisponde a 28.375 grammi.

(Le risposte agli Esercizi sono date alla fine di ciascuna Unità insieme alle risposte ai TEST.)

Esercizio 2

Ogni conversione ha come base la relazione 'fattore di conversione × numero da convertire' così è possibile scrivere un programma generale di conversione nel quale inserite due numeri ogni volta che lo usate: il fattore di conversione ed il numero da convertire. Scrivete un programma generale di conversione che faccia tutto questo.

1.13 Istruzione di commento: REM

L'istruzione REM è una istruzione di commento. Essa ci consente di dare un titolo al programma e di inserirvi alcune annotazioni importanti. Ad esempio, inserita all'interno del programma ci aiuta a identificare la funzione del programma o di una parte di esso. L'istruzione REM non viene eseguita dal computer ed è presente unicamente a beneficio del programmatore o dell'utilizzatore, vale a dire che quando il computer vede REM all'inizio di una linea ignora tutto ciò che si trova nella linea stessa.Il prossimo programma riguarda il calcolo delle percentuali e come titolo del programma la nostra prima istruzione sarà 10 REM **CALCOLO DI PERCENTUALI** . I due asterischi (**) non hanno altra funzione che evidenziare il titolo. Noi possiamo anche usare REM per spiegare delle parti di programma che possono essere difficili da comprendere, come vedrete più avanti.

Esempio 3

Scriviamo un programma BASIC per inserire due numeri e ricevere come risultato un numero che rappresenta la percentuale del secondo rispetto al primo.

Ricordate che Percentuale=(secondo/primo)×100

Soluzione

```
10 REM **CALCOLO PERCENTUALE**
20 INPUT F
30 INPUT S
40 LET P=(S/F)*100
50 PRINT P
60 STOP
```

Programma 12 Calcolo percentuale

Funzionamenti tipo

```
RUN
L 57
L 74
129.82456

9/60
```

```
RUN
L 74
L 57
77.027027

9/60
```

K Programma 12

1.14 Aritmetica più complessa

Noi siamo giunti allo stadio nel quale possiamo utilizzare il computer come una semplice macchina calcolatrice a quattro funzioni, ma noi vorremo presto fare dei calcoli aritmetici un po più complicati. In generale il BASIC ci consente di esporre delle equazioni in modo molto familiare. Noi possiamo usare delle parentesi, come (), per raggruppare insieme certi valori, e quando il BASIC valuta un'espressione tratta per primi i valori all'interno delle parentesi. Quindi vengono i valori che comportano moltiplicazioni e divisioni ed alla fine, addizioni e sottrazioni.

Questo ordine di preferenza nel compiere le operazioni aritmetiche è discusso più ampiamente in una prossima Unità, ma vedrete presto che questo ordine formalizza soltanto la strada che normalmente seguiamo nel fare dei calcoli aritmetici. Vi esponiamo quello che intendiamo dire.

Esempio 4

Scriviamo le seguenti espressioni in BASIC:

1. $ab+c$ 2. $a(b+c)$ 3. $\frac{a}{b+c}$

Vedi Nota 3)

1. A*B+C
   Le regole sull'ordine di precedenza ci dicono che A*B sarà valutata per prima e che quindi verrà aggiunto C. Se voi siete infastiditi da questo potete scrivere (A*B) + C ma le parentesi non sono indispensabili in questo caso.
2. A*(B+C)
   Notate che, come le parentesi sono necessarie in A(B+C) così sono necessarie in A*(B+C).
3. A/(B+C)

Ora provate da soli.

TEST 8

Scrivete le seguenti espressioni in BASIC:

1. abc

2. $\frac{ab}{c}$

3. $\frac{a+b}{c}$

Esercizio 3

Ora che avete scritto le espressioni del TEST 8 come espressioni BASIC, scrivete un programma che vi consenta di inserire tre numeri (A,B,C) e di ritornare i valori delle espressioni del TEST 8.

1.15 Stampa di caratteri alfabetici

Avete già visto che possiamo visualizzare i valori delle locazioni di memoria. Scoprirete, come il corso procede che l'istruzione PRINT è molto versatile. Uno degli usi di questa istruzione è quello di visualizzare dei messaggi sullo schermo del monitor, il che è di aiuto all'utilizzatore quando il programma è in funzione. A questi messaggi viene fatto usualmente riferimento come prompts (messaggi di 'pronto'). Noi abbiamo già visto che quando, durante l'esecuzione del programma, si incontra una istruzione di input, appare sullo schermo una L lampeggiante per ricordarci che è richiesto l'inserimento di un dato. In alcuni programmi un po più complicati, una serie di L sullo schermo può confondere l'utilizzatore poiché egli può non conoscere che tipo di valore viene richiesto dal prompt L. I messaggi di pronto prodotti con l'istruzione PRINT sono molto utili in queste circostanze.

È molto facile ottenere dal computer di visualizzare un promemoria o un messaggio sullo schermo. Tutto ciò di cui avete bisogno è una linea del tipo:

```
20 PRINT "MESSAGGIO"
```

E questa semplicemente stampa

```
MESSAGGIO
```

sul vostro schermo.

In altre parole, tutto ciò che appare tra gli apici " " dopo la parola PRINT sarà stampato esattamente come si trova. Notate che, come nel caso dell'istruzione REM, il computer non esegue come istruzioni le parole all'interno degli apici. Per cui

```
20 PRINT "A+B"
```

fa apparire

```
A+B
```

sullo schermo ma il computer non addiziona il valore nella locazione A al valore nella locazione B.

Il seguente esempio dimostra l'uso di PRINT " " per ricordare al programmatore e all'utilizzatore cosa il programma sta per fare.

Esempio 5

Scriviamo un programma BASIC per convertire un valore di temperatura dato in gradi C in gradi F.

Ricordate che $°F = \frac{9}{5} °C + 32$

Soluzione

```
10 REM **CONVERSIONE GRADI
         CENTIGRADI IN
         FAHRENHEIT**
20 PRINT "INSERIRE TEMPERATURA
IN GRADI C"
30 INPUT C
40 PRINT C
50 LET F=(9/5)*C+32
60 PRINT "QUESTA TEMP. IN GRAD
I F SONO"
70 PRINT F
80 STOP
```

Il messaggio in Print precede l'istruzione di Input in modo che il messaggio venga visualizzato prima che appaia il prompt L

Programma 13 Conversione di temperatura

Funzionamento tipo

```
RUN
INSERIRE TEMPERATURA IN GRADI C
16
QUESTA TEMP. IN GRADI F SONO
60.8
```

K Programma 13.

Obbiettivi del capitolo 1

Ora che avete completato questa Unità, verificate che siete in grado di:

Scrivere semplici programmi usando:

Numeri di riga ☐

INPUT ☐

LET ☐

PRINT ☐

Locazioni di memoria identificate da una singola lettera o da una lettera seguita da una singola cifra ☐
La copiatura da una locazione ad un'altra ☐
La sovrascrittura ☐
+,-,*,/ ☐
( ) ☐
Costanti numeriche ☐
REM ☐
PRINT " " ☐
Sapere quando usare:
NEW LINE ☐
RUN ☐

Sapere come rispondere a:

Cursori L e K ☐

Segnale di prompt L ☐

Risposte ai TEST ed agli Esercizi

TEST 1

A	B
1	(c)
2	(e)
3	(g)
4	(a)
5	(b)
6	(f)
7	(d)

TEST 2

(a) e (e) funzionano come il programma 1.
(b) è chiamato a stampare la SOMMA prima che sia calcolata.
(c) e (d) sono chiamati a calcolare la SOMMA prima che venga inserito il SECONDO numero.

TEST 3

(Avete notato che questo programma ha usato correttamente numeri decimali frazionari e negativi?)

TEST 4

(a) OK
(b) No, comincia con un numero invece di una lettera.
(c) Non disponibile su tutti i sistemi. (10 è di due cifre, non come in W8.)
(d) No, # non è un simbolo accettabile nel nome di una variabile.
(e) No, usa due lettere. (Questo può, naturalmente, andar bene su alcune macchine.)
(f) OK
(g) OK
(h) No, '-' è un simbolo non accettabile.
(i) No, '+' è un simbolo non accettabile.
(j) OK

TEST 5

(a) 3+7 (b) 3*7 (c) 8/4 (d) 5*(2+8) (e) 30/(3+2)
(f) 24-(4*3) (g) 5*6*7 (h) 81-(27*2)

TEST 6

(a) 17 (b) 10 (c) 100 (d) 5 (e) 10/3 o 3.33 ...
(f) 4 (g) 50/3 o 16.66 ... (h) 75

TEST 7

Esercizio 1

(a) Programma 14 Funzionamenti tipo

(b) Programma 15

```
 5 REM **PROGRAMMA 16**
10 INPUT W1
20 LET W2=W1*28.375
30 PRINT W2
40 STOP
```

Funzionamenti tipo

K Programma 15.

Esercizio 2

Programma 16

Funzionamenti tipo

K Programma 16.

TEST 8

1. A*B*C
2. A*B/C [(A*B)/C è anche corretto, sebbene le parentesi non siano necessarie].
3. (A+B)/C

Esercizio 3

Programma 17

```
  5 REM **PROGRAMMA 17**
 10 INPUT A
 20 INPUT B
 30 INPUT C
 40 LET R=A*B*C
 50 PRINT R
 60 LET R=(A*B)/C
 70 PRINT R
 80 LET R=(A+B)/C
 90 PRINT R
100 STOP
```

Righe 40–50: Calcola e stampa la prima espressione
Righe 60–70: Calcola e stampa la seconda espressione
Righe 80–90: Calcola e stampa la terza espressione

Notate che possiamo usare 'R' come locazione per tutti e tre i risultati perché noi copiamo (stampiamo) a turno ciascuna risposta prima di sovrascrivere la successiva risposta.

Funzionamenti tipo

```
RUN                     RUN
L 13                    L 13
L -27                   L 13
L 55.2                  L 13
-19375.2                2197
-6.3586957              13
-0.25362319             2

9/100                   9/100
```

K Programma 17

CAPITOLO 2

PRENDERE LE DECISIONI

2.1 Introduzione

I programmi presi in esame nell'Unità 1 erano abbastanza lineari. Essi iniziavano l'esecuzione dalla istruzione con il numero di linea più basso e continuavano in ordine di numero di linea finché l'esecuzione terminava alla linea con il numero di linea più alto. Una cosa nella quale i computers sono molto efficaci e nei calcoli ripetitivi; un'altra è la loro capacità di prendere delle decisioni. Entrambe queste prestazioni richiedono il cambiamento della sequenza nella quale un programma viene eseguito. Questa Unità vi introdurrà ad alcune delle istruzioni che vi consentono di scrivere programmi di questo tipo. Ma per prima cosa vorremmo introdurvi ad un nuovo tipo di istruzione PRINT.

2.2 PRINT...,

Nella Unità 1 scrivemmo un programma (Esempio 3) per ottenere un numero come percentuale di un'altro numero. Sullo schermo, il calcolo ed il risultato apparivano nel formato:

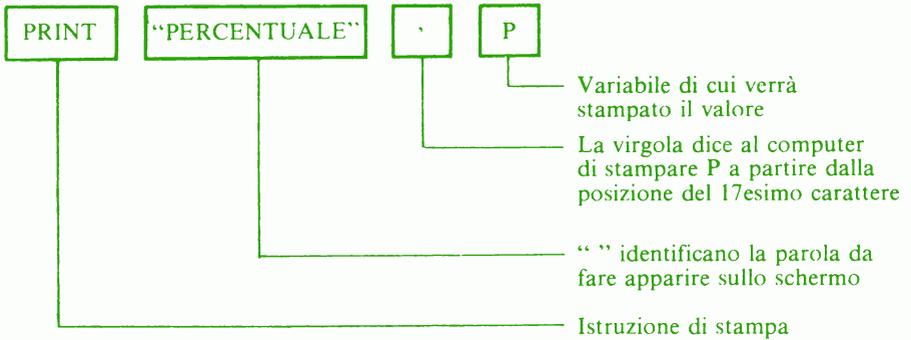
```
RUN
 L  57
 L  74
129.825
```

Tuttavia sarebbe meglio se la risposta includesse la parola 'Percentuale' in modo che fosse chiaro cosa avviene. Questo può essere fatto facilmente cambiando la linea 50 da 50 PRINT P a

```
50 PRINT "PERCENTUALE",P
```

L'effetto di questo è:

Versione della linea 50	Risultato sullo schermo
50 PRINT P	129.825
50 PRINT P	PERCENTUALE 129.825

L'istruzione PRINT "PERCENTUALE",P è formata da quattro parti.

Possiamo usare PRINT..., per migliorare il programma il programma delle percentuali dalla Unità 1. Allo stesso tempo possiamo migliorare l'aspetto del programma facendo uso della istruzione di stampa di caratteri alfabetici PRINT " " che abbiamo introdotto nella sezione 1.15.

Programma originale

```
10 REM **VALCOLO PERCENTUALE**
20 INPUT F
30 PRINT F
40 INPUT S
50 PRINT S
60 LET P=(S/F)*100
70 PRINT P
80 STOP
```

Nuovo programma

```
  10 REM **CALCOLO PERCENTUALE**
  15 PRINT "INSERIRE IL PRIMO NU
MERO"
  20 INPUT F
  30 PRINT F
  35 PRINT "INSERIRE IL SECONDO
NUMERO"
  40 INPUT S
  50 PRINT S
  60 LET P=(S/F)*100
  70 PRINT "PERCENTUALE",P
  80 STOP
```

Programma 1 Programma percentuali migliorato

Programma originale: funzionamento tipo

```
RUN
L 80
L 37
46.25
```

Nuovo programma: funzionamento tipo

```
INSERIRE IL PRIMO NUMERO ———effetto della riga 15
80
INSERIRE IL SECONDO NUMERO ——effetto della riga 35
37
PERCENTUALE      46.25 —————effetto della riga 70
```

Notate come l'aggiunta delle righe di stampa 15 e 35, insieme a PRINT " ", ed alla stampa dei dati inseriti rendano l'uso del programma molto più comprensibile ed 'amichevole'.

K Programma 1

Aree di stampa

Ci sono due aree di stampa sullo schermo di uno ZX81. L'Area 1 è alla sinistra dello schermo; l'Area 2 è invece ottenuta tramite l'uso della virgola nell'istruzione PRINT. Quindi

```
PRINT "AREA 1",          "AREA 2"
```

dà come risultato

```
AREA 1          AREA 2
```

e

```
PRINT "PERCENTUALE",P
```

dava

```
PERCENTUALE 46.25
```

mentre

```
PRINT "PERCENTUALE";P
```

dovrebbe dare

```
PERCENTUALE46.25
```

TEST 1

Come dovrebbe essere il risultato sullo schermo delle seguenti linee di stampa?(Stabilendo i valori di A=48, B=8, C=6).

(a) PRINT "AREA";A
(b) PRINT "LUNGH.";B,"LARGH.";C
(c) PRINT "LUNGH.","LARGH."
PRINT B,C

Scrivete le linee PRINT in BASIC che dovrebbero stampare sullo schermo le seguenti parole nelle rispettive aree:

	Area 1	Area 2
(d)	LUNGH.	8
(e)	LUNGH.	8
	LARGH.	6
	AREA	48
(f)	LUNGH.	LARGH.

2.3 Ripetizione e GO TO

Supponete ora di aver bisogno di usare il Programma 1 per calcolare tutte le percentuali di un esercizio svolto da una intera classe di studenti. Voi dovrete usare il programma diverse volte, iniziando ogni volta con RUN, per es.

```
RUN
INSERIRE IL PRIMO NUMERO
80
INSERIRE IL SECONDO NUMERO
42
PERCENTUALE      52.5
```

```
RUN
INSERIRE IL PRIMO NUMERO
80
INSERIRE IL SECONDO NUMERO
19
PERCENTUALE      23.75
```

Figura 1 Uso ripetuto del programma percentuali

Sarebbe molto più semplice se, dopo che il computer ha calcolato la prima percentuale, tornasse indietro all'inizio del suo calcolo e ci chiedesse di inserire i prossimi dati. Possiamo far questo facendo uso della istruzione:

'GO TO numero di riga'

che dirige il programma verso qualsiasi numero di riga inserito. Qui di seguito vi è il programma delle percentuali riscritto in questo modo:

```
  10 REM **PERCENTUALI**
  30 PRINT "INSRERIRE IL VOTO MA
SSIMO"
  40 INPUT T
  50 PRINT T
  70 PRINT "INSERIRE IL PROSSIMO
 VOTO"
  80 INPUT M
  90 PRINT M
 110 IF M=-9999 THEN GOTO 180
 130 LET P=(M/T)*100
 140 PRINT "PERCENTUALE",P
 160 GOTO 70
 180 STOP
```

Programma 2 Programma percentuali per uso ripetuto

Notate le nuove righe 20 e 30 che ci permettono di inserire il valore massimo del voto una sola volta durante l'uso del programma. Il calcolo viene effettuato con le righe da 50 a 100 e, quando il computer legge la riga 110, il programma ritorna alla riga 50 per chiederci l'inserimento di un'altro voto. Un tipico funzionamento può essere:

RUN

```
INSRERIRE IL VOTO MASSIMO
80
INSERIRE IL PROSSIMO VOTO
42
PERCENTUALE      52.5
INSERIRE IL PROSSIMO VOTO
67
PERCENTUALE      83.75
INSERIRE IL PROSSIMO VOTO
```

Ricordate che, se lo schermo è pieno, allora occorre premere CONT per andare avanti con il programma.

L'istruzione GO TO interrompe la normale esecuzione del programma in ordine di numero di riga. Appena il programma incontra un GO TO trasferisce incondizionatamente il controllo alla riga il cui numero è specificato nell'istruzione. Questo alcune volte chiamato un 'salto incondizionato'.

K Programma 2 (Premete BREAK e quindi NEW LINE per interrompere l'esecuzione del programma.)

TEST 2

Il seguente programma fornisce il quadrato di un numero (moltiplica un numero per se stesso). Aggiungete una riga con GO TO in modo da consentire di usare il programma più volte per calcolare il quadrato di una serie di numeri.

```
 5 REM **CALCOLO QUADRATI**
10 INPUT N
20 LET S=N*N
30 PRINT S
40 GOTO 5
50 STOP
```

Programma 3 Calcolo del quadrato

K inserite e fate girare le righe da 5 a 40. Aggiungete la riga con GO TO e fate girare il vostro nuovo programma.

2.4 Stile di programmazione

L'uso dell'istruzione GO TO è di aiuto in molti casi. Tuttavia, essa lascia in sospeso alcune cose, come la [L] lampeggiante alla fine dell'esecuzione. All'arrivo della linea 110 nel Programma 2 il controllo è sempre ritornato alla linea 50 che quindi produce la richiesta di un nuovo input di un dato e quindi la L . Il programma risulta bloccato in un circolo continuo dal quale non può uscire. La sola via per fermare questo programma è di interromperlo premendo STOP.

Avere una esecuzione del programma che rimane a mezz'aria è ovviamente un cattivo stile, ma elimineremo questo difetto in breve tempo. Più importante è però avvertirvi dei pericoli dell'uso di GO TO. Come abbiamo detto, questa istruzione permette di saltare verso qualsiasi punto nel programma, il che può sembrare una prestazione utile; poiché GO TO ci consente di saltare abbastanza a caso verso qualsiasi punto del programma, essa è spesso usata in modo tale che la struttura logica della soluzione sia interrotta da 'salti di convenienza' verso altre parti del programma, piuttosto che seguendo la struttura logica dell'analisi del problema. Noi, perciò, useremo l'istruzione GO TO moderatamente all'interno di questo corso; la useremo solo quando pensiamo che la chiarezza del programma risulti compromessa dal suo mancato uso. Speriamo che anche voi seguiate il nostro esempio ed usiate l'istruzione GO TO il meno possibile. Anche se evitiamo un suo uso indiscriminato, la troverete più largamente utilizzata in alcuni libri di testo e riviste di computers.

2.5 IF ... THEN GO TO ...

Il problema che abbiamo appena lasciato è come segnalare al computer che abbiamo raggiunto la fine della lista di voti. Tra non molto vedremo in che modo il computer può essere usato per contare al nostro posto, per prima cosa però introdurremo un sistema per segnalare che si è raggiunta la fine della lista.

Un metodo è quello di terminare la lista dei numeri inseriti con un numero speciale che consenta di uscire dal programma, per es. -9999. Noi possiamo difficilmente aspettarci che uno degli studenti possa ottenere il risultato di -9999 in un test, questo valore viene chiamato 'valore finale'. Noi abbiamo bisogno che il programma funzioni normalmente quando vengono inseriti dei valori 'propri' ma che si fermi quando viene inserito il voto -9999. In altre parole, dobbiamo essere in grado di scrivere un programma con la seguente struttura logica:

1. Inizio
2. Inserire il voto massimo.
3. Inserire il prossimo voto.
4. Se (IF) questo valore è uguale a -9999 allora (THEN) vai alla (GO TO) riga 8 altrimenti continua con la riga 5.
5. Calcolo della percentuale.
6. Stampa la percentuale.
7. Vai alla (GO TO) riga 3.
8. Fine.

Figura 2 Interruzione del calcolo percentuale

Fortunatamente nel BASIC c'è un'istruzione che può eseguire la decisione nella riga 4. Essa è:

```
IF ... THEN GO TO numero di riga
   |
   └──── condizione che deve verificarsi per avere un salto del programma
         alla riga specificata
```

Così tutto quello che dobbiamo fare è di tradurre l'istruzione 4 in Figura 2 come

```
110 IF M=-9999 THEN GO TO 180
```

e farla diventare parte del programma 4. Questa istruzione significa: se il valore trovato in M è uguale a -9999, allora vai alla riga 180, altrimenti continua eseguendo l'istruzione seguente dopo 110. L'istruzione in figura 2 'altrimenti continua con la riga 5' non viene tradotta in BASIC ma è implicita. Il programma con la nuova riga può essere

```
  10 REM **PERCENTUALI**
  30 PRINT "INSERIRE IL VOTO MAS
SIMO"
  40 INPUT T
  50 PRINT T
  70 PRINT "INSERIRE IL PROSSIMO
 VOTO"
  80 INPUT M
  90 PRINT M
 110 IF M=-9999 THEN GOTO 180
 130 LET P=(M/T)*100
 140 PRINT "PERCENTUALE",P
 160 GOTO 70
 180 STOP
```

Programma 4 Calcolo percentuale con valore finale

Usate il Programma 4 esattamente nello stesso modo del Programma 2 finché non avete inserito l'ultimo voto da convertire in percentuale. Quindi inserite il voto -9999 ed il programma termina.

Un funzionamento tipo è

```
RUN
INSERIRE IL VOTO MASSIMO
80
INSERIRE IL PROSSIMO VOTO
43
PERCENTUALE       53.75
INSERIRE IL PROSSIMO VOTO
29
PERCENTUALE       36.25
INSERIRE IL PROSSIMO VOTO
62
PERCENTUALE       77.5
INSERIRE IL PROSSIMO VOTO
-9999 ——————————————— voto finale

9/180
```

K Programma 4 usatelo per convertire da soli dei voti in percentuale e concludete con -9999.

TEST 3

Avete finito il TEST 2 con il programma:

```
 5 REM **CALCOLO QUADRATI**
10 INPUT N
20 LET S=N*N
30 PRINT S
40 GOTO 5
50 STOP
```

ma come per il programma delle percentuali, anche questo non si ferma mai. Modificate il programma includendovi un valore finale che permetta di uscire dal ciclo.

2.6 Ineguaglianze

È di aiuto potere usare l'espressione M=-9999 nell'istruzione IF ... THEN GO TO ... per stabilire quando deve o non deve essere effettuato un salto nel programma. L'istruzione significa se (IF) M=-9999 è vero, allora (THEN) salta alla linea indicata, altrimenti continua. Il segno '=' stabilisce una relazione tra M e -9999.

Il BASIC consente alle espressioni di includere delle relazioni:

Relazione	Esempio	Significato
>	A>B	Il valore nella locazione A è maggiore del valore nella locazione B
<	X<Y	Il valore nella locazione X è minore del valore nella locazione Y

Vero o falso?

Considerate l'espressione A<B. Se A=2 e B=5 allora A<B è vero, perché 2 è minore di 5. Considerate la stessa espressione con i valori di A=2 e B=1. Ora A<B è falso, perché 2 non è minore di 1. Allo stesso modo se A=2 e B=2 A<B è falso, in quanto 2 non è minore di 2. Nello scrivere programmi spesso troveremo utile essere in grado di conoscere se una istruzione che coinvolge = o < > sia vera o falsa. Questo è chiamato lo stato logico della asserzione, per es.

Asserzione	Stato logico
3>2	Vero
7<7	Falso

Voi probabilmente troverete ciò abbastanza facile con i numeri interi positivi ma potreste essere meno sicuri di cosa accada in altri casi. Se siete in dubbio, ricordate la linea dei numeri:

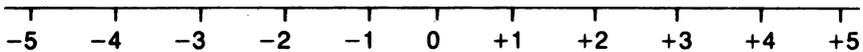



Figura 3 La linea dei numeri

Se un numero si trova ad essere su questa linea alla sinistra di un secondo numero, allora il primo numero è minore del secondo; se invece si trova alla destra allora il primo numero è maggiore del secondo.

Esempio 1

Verificate quali delle seguenti espressioni sono vere o false con i valori dati per A e B. F = falso V = vero

Valori		Espressione
A	B	
2	5	A>B
2	—5	A>B
—2	—5	A>B
—2	—1	A>B
—5	2	A<B
5	—2	A<B
—3	3	A=B

Soluzione

Per far questo elaboriamo i valori di ciascuna espressione utilizzando quelli dati ed usiamo la linea dei numeri per decidere se l'asserzione è vera o falsa per quei particolari valori. Così la soluzione è:

Valori		Asserzione		
A	B	Espressione	Suo valore	Suo stato logico
2	5	A>B	2> 5	F
2	—5	A>B	2>—5	V
—2	—5	A>B	—2>—5	V
—2	—1	A>B	—2>—1	F
—5	2	A<B	—5< 2	V
5	—2	A<B	5<—2	F
—3	3	A=B	—3= 3	F

F = falso V = vero

TEST 4

Completate la seguente tabella per determinare se le espressioni date sono vere o false con i valori indicati.

Valori		Asserzione		
A	B	Espressione	Suo valore	Suo stato logico
3	7	A>B		
5	3	A>B		
—3	5	A>B		
8	5	A<B		
3	9	A<B		
8	—2	A<B		

Siamo ora in condizione di usare le relazioni per consentire al programma di saltare ad una nuova riga quando vengono soddisfatte certe condizioni.

Esempio 2

Nel seguente segmento di programma, dopo l'esecuzione della riga 30, il controllo dovrebbe passare alla riga 40 o alla riga 100?

```
10 LET A=3
20 LET B=2
30 IF A+B>0 THEN GOTO 100
```

Programma 5

Soluzione

L'espressione $-3+2>0$ è falsa, per cui il salto alla linea 100 non avviene ed il controllo passerà alla linea 40.

TEST 5

Nei seguenti segmenti di programma, dopo l'esecuzione della linea 30, il controllo dovrebbe passare alla linea 40 o alla linea 100?

(a)
```
10 LET A=7
20 LET B=-8
30 IF A-B<0 THEN GOTO 100
```

(b)
```
10 LET X=3
20 LET Y=-3
30 IF X/Y=-1 THEN GOTO 100
```

(c)
```
10 LET P=-1
20 LET Q=3
30 IF P+Q>Q THEN GOTO 100
```

(d)
```
10 LET M=3
15 LET N=-4
20 LET P=-2
30 IF M-N<N-P THEN GOTO 100
```

(e)
```
10 LET R=1
20 LET S=-2
30 IF R+S>-1 THEN GOTO 100
```

Programmi 6-10

2.7 Diagrammi di flusso

Come abbiamo detto, il compito principale di un programmatore è quello di trovare la strada più conveniente per esprimere la soluzione di un particolare problema. A questo punto dobbiamo introdurvi a ciò che dovrebbe essere la peggiore parola nel gergo dei computer: algoritmo. Questa parola è usata per indicare una strategia risolutiva generale, ed è definita come una serie di istruzioni o passi procedurali per la soluzione di un problema specifico. Noterete che in questo caso il computer non è menzionato. A parte ciò, le definizioni di programma ed algoritmo sono identiche. Un programma, quindi, è un algoritmo scritto per un computer.

Ci sono tre modi fondamentali di enunciare un algoritmo:

(i) una descrizione
(ii) dei codici BASIC
(iii) un diagramma di flusso

I diagrammi di flusso sono un pò come dei progetti e sono di aiuto a chi preferisce vedere gli eventi visualizzati in forma di illustrazione o di grafico.

Visualizziamo le differenti funzioni all'interno di un algoritmo usando dei riquadri di forma diversa.

Una —►— indica la sequenza dell'algoritmo, mentre i riquadri contengono delle scritte appropriate.

Il primo programma della Unità 1 può essere espresso in forma di diagramma di flusso nel modo seguente:

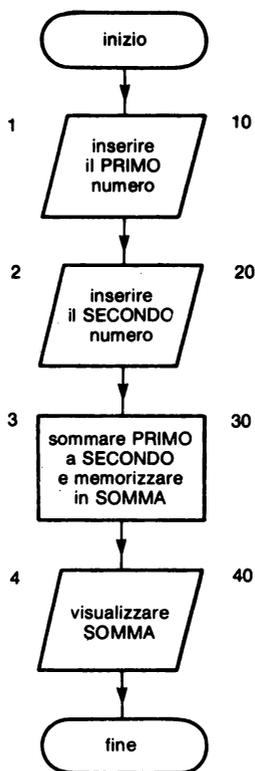



Figura 4 Diagramma di flusso del Programma 1 dall'Unità 1

Le descrizioni delle funzioni nei riquadri sono brevi ma dovrebbero essere comprese anche da chi non ha conoscenza del BASIC. A questo riguardo diciamo che cerchiamo di tenere queste desrizioni indipendenti dal linguaggio BASIC. I numeri sul lato sinistro dei riquadri si riferiscono alle istruzioni come vengono descritte nella Figura 4 dell'Unità 1, mentre i numeri alla destra dei riquadri si riferiscono alle istruzioni BASIC del Programma 1 dell'Unità 1.

TEST 6

Costruite il diagramma di flusso del programma sulle percentuali (Programma 1).

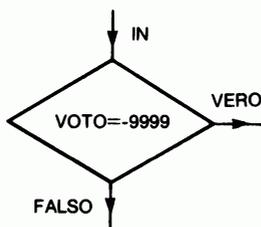
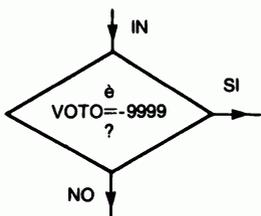
I riquadri di decisione

Voi avete visto come le decisioni sono effettuate in BASIC usando l'istruzione IF ... THEN ... Il significato logico e:

IF l'affermazione è vera THEN vai alla riga X
altrimenti (affermazione falsa) passa alla prossima riga

L'idea base di saltare alla riga X o di continuare con la riga successiva è rappresentata nel diagramma di flusso da due linee che fuoriescono da un riquadro di decisione. La decisione della linea 110 nel Programma 4:

```
110 IF M=-9999 THEN GO TO 180
```

potrebbe essere descritta nella forma

L'affemazione VOTO= -9999 può essere espressa come una domanda

Lo stile di un diagramma di flusso dipende da voi. La prova dell'efficacia di un diagramma di flusso è se riuscite o no a seguire facilmente il diagramma qualche tempo dopo averlo realizzato pure, se voi provate a comunicare le vostre idee ad altri, ed essi riescono a seguire facilmente il vostro diagramma di flusso.

Un diagramma di flusso per il Programma 4 viene dato qui di seguito:

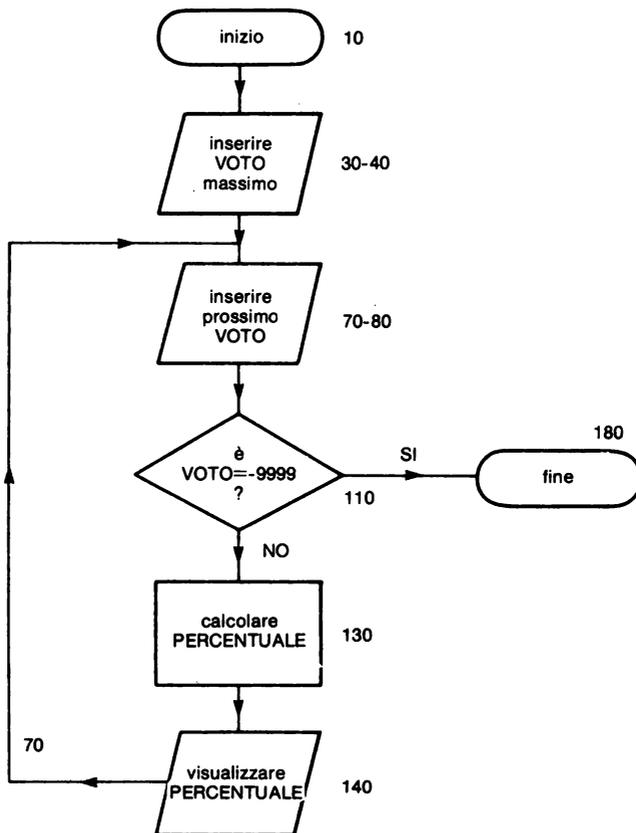



Figura 5 Diagramma di flusso per il programma delle percentuali

I numeri alla destra dei riquadri si riferiscono ai numeri di riga delle istruzioni nel programma. L'istruzione 160 GO TO 70 è rappresentata dalla riga che ritorna indietro al riquadro con l'istruzione 70.

TEST 7

Scrivete un diagramma di flusso per il programma che avete scritto per rispondere al TEST 3.

Ora che abbiamo introdotto il concetto di diagrammi di flusso, possiamo usarli per aiutarci a progettare la struttura dei programmi che scriveremo.

2.8 Contare

Come abbiamo detto, i computers sono ottimi nello svolgere delle procedure ripetitive. Se, inoltre, vogliamo controllare queste attività invece che soltanto iniziarle e finirle come abbiamo fatto nell'ultimo esempio, dobbiamo usare il computer per contare le ripetizioni al nostro posto. Se stabiliamo uno specifico numero di ripetizioni di una attività, possiamo iniziare a contare alla prima attività e, aggiungere uno per ogni successiva attività, finché non arriviamo ad un limite prestabilito. Possiamo descrivere questa procedura in forma di diagramma di flusso.

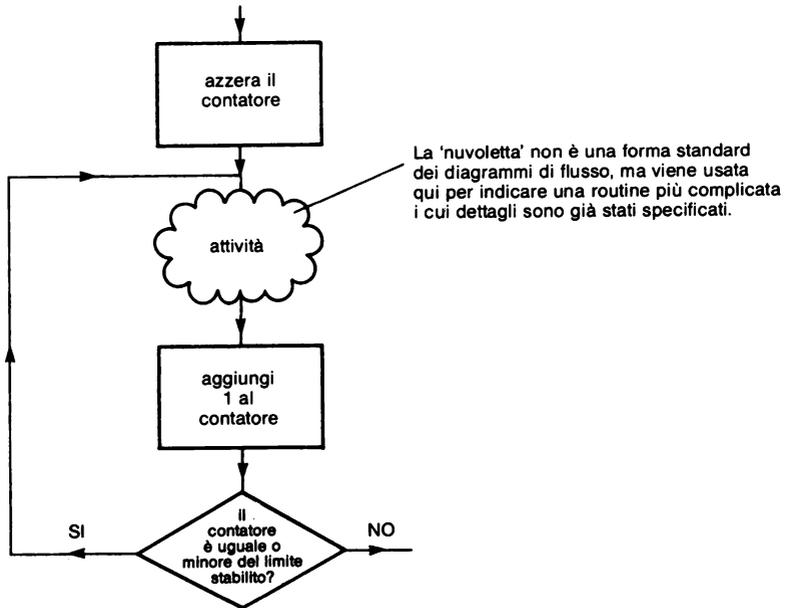



Figura 6 Un contatore in un diagramma di flusso

Notate che il contatore è formato da tre parti:

(i) la procedura che dà al contatore il suo valore iniziale;
(ii) la procedura che aggiunge 1 al contatore ogni volta che l'attività è completata;
(iii) la procedura per fermare il conteggio ed abbandonare l'attività dopo che essa è stata eseguita per il numero di volte richiesto.

TEST 8

Quanti numeri saranno inseriti con i programmi descritti dai seguenti diagrammi di flusso?

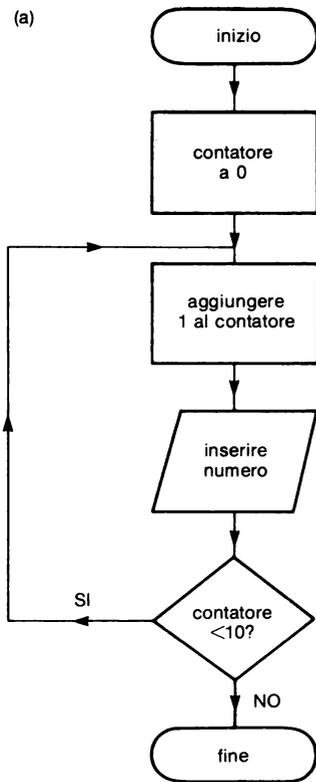
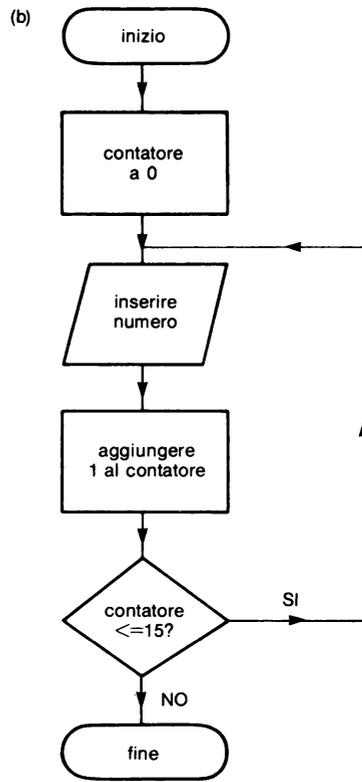



Figura 7a Figura 7b

TEST 9

Completate l'istruzione IF ... THEN GO TO ... del seguente programma in modo che si possano inserire 5 numeri.

```
 5 REM **TEST 9**
10 LET C=0
20 INPUT N
30 IF C=B THEN GOTO 55
40 LET C=C+1
50 GOTO 20
60 STOP
```

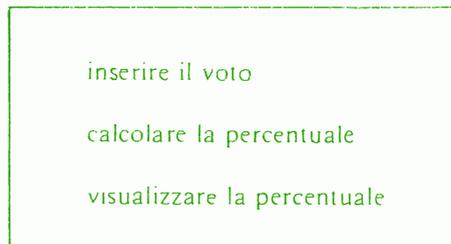
Programma 11

Esempio 3

Scriviamo un programma BASIC per calcolare e visualizzare il voto in percentuale per un gruppo di cinque studenti.

Soluzione

Se stabiliamo che la 'attività' nella nuvoletta nel diagramma di flusso di Figura 6 sia:

inserire il voto calcolare la percentuale visualizzare la percentuale

possiamo esporre l'algoritmo della soluzione in forma di diagramma di flusso.

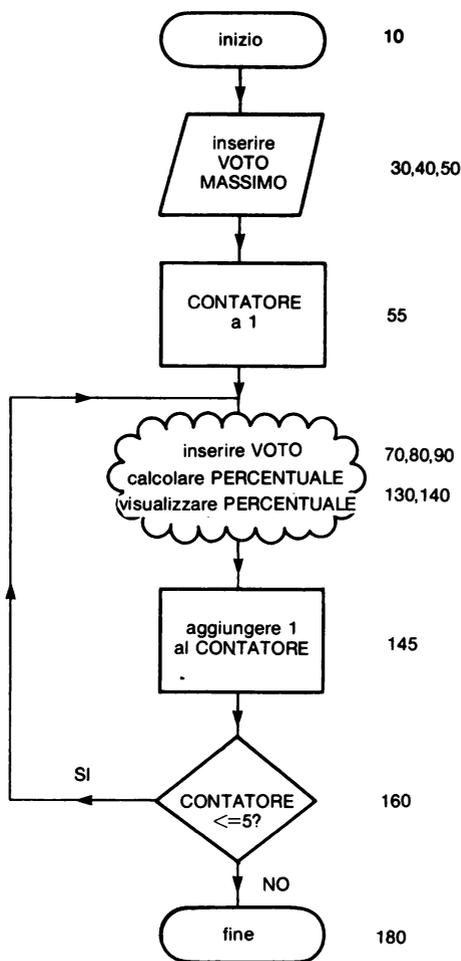



Figura 8 Diagramma di flusso per il calcolo della percentuale su 5 voti

Questo ci indica la struttura del programma che dobbiamo scrivere. Il programma attuale può essere ora scritto modificando il Programma 4 per aggiungere il contatore. Così il programma richiesto è:

```
10 REM **PERCENTUALI**

  30 PRINT "INSERIRE IL VOTO
MASSIMO"
  40 INPUT T
  50 PRINT T
  55 LET C=1
  70 PRINT "INSERIRE
IL PROSSIMO VOTO"
  80 INPUT M
  90 PRINT M
 130 LET P=(M/T)*100
 140 PRINT "PERCENTUALE",P
 145 LET C=C+1
 150 PRINT "LINEA 150",
"CONTEGGIO QUI= ";C
 160 IF C<=5 THEN GOTO 70
 180 STOP
```

partenza del contatore 'inizializzazione' (linea 55)

aggiunge 1 al contatore 'incremento' (linea 145)

abbiamo aggiunto questa linea per controllare cosa appare nel contatore a questo punto del programma. (linea 150)

conteggio completato? (linea 160)

Programma 12 Aggiunta di un contatore al programma delle percentuali

RUN

```
INSERIRE IL VOTO MASSIMO
75
INSERIRE IL PROSSIMO VOTO
57
PERCENTUALE      76
LINEA 150        CONTEGGIO QUI= 2
INSERIRE IL PROSSIMO VOTO
62
PERCENTUALE      82.666667
LINEA 150        CONTEGGIO QUI= 3
INSERIRE IL PROSSIMO VOTO
43
PERCENTUALE      57.333333
LINEA 150        CONTEGGIO QUI= 4
INSERIRE IL PROSSIMO VOTO
39
PERCENTUALE      52
LINEA 150        CONTEGGIO QUI= 5
INSERIRE IL PROSSIMO VOTO
70
PERCENTUALE      93.333333
LINEA 150        CONTEGGIO QUI= 6
```

Nota: uscendo dal ciclo il valore del contatore è 6.

K Programma 12.

Esercizio 1

Considerate il problema di stimare il costo dell'istallazione di una finestra in vetro con infissi di alluminio. La finestra comprende tre parti:

(a) un bordo di legno,
(b) un telaio di alluminio ed
(c) il vetro.

Se l'altezza della finestra è di H metri e la larghezza è di W metri, allora la lunghezza totale del legno e dell'alluminio occorrente è data approssimativamente dalla espressione:

$$(2H+2W)$$

metri; e l'area del vetro occorrente, dalla espressione:

$$(HxW)$$

metri quadrati.

Realizzate un programma BASIC per calcolare il costo di sei finestre a partire dall'inserimento dei valori di altezza (H) e larghezza (W) di ciascuna finestra e tenendo presente che il costo dei materiali è di L.7500 al metro per il bordo in legno, di L.10000 al metro per il telaio di alluminio e di L.100000 al metro quadrato per il vetro e che il costo del lavoro è di L.125.000 per ogni finestra.

Notate che la 'attività' da ripetere sarà:

inserire altezza e larghezza della finestra calcolare il costo dell'istallazione di questa finestra visualizzare il costo della finestra

Procedete nel modo seguente:

(a) Disegnate un algoritmo in forma di diagramma di flusso per calcolare e visualizzare il costo di ciascuna delle sei finestre.

(b) Convertite l'algoritmo in istruzioni BASIC. Inserite il programma nel vostro microcomputer e provatelo calcolando il costo di sei finestre.
(c) Perfezionate il vostro programma per calcolare il costo di un numero qualsiasi di finestre, da specificare all'inizio del programma.

2.9 Comparazione

Noi abbiamo visto in che modo il linguaggio BASIC ci permette di comparare due numeri. Spesso noi vogliamo stabilire se un particolare valore è più grande o più piccolo di un'altro. Questo è un processo fondamentale per riordinare e selezionare dei dati. Durante il corso prenderemo in considerazione i metodi di riordinamento in dettaglio, così iniziamo con il caso più semplice.

Esempio 4

Realizziamo un algoritmo in forma descrittiva per inserire due numeri e visualizzare il più grande dei due.

Commento

Noi abbiamo espresso i nostri algoritmi come diagrammi di flusso durante gran parte di questa unità così questa volta useremo il metodo descrittivo introdotto nell'Unità 1.

Soluzione

1. Inizio.
2. Inserire il primo numero.
3. Inserire il secondo numero.

4. Se il primo numero è > del secondo numero allora vai alla linea 7 altrimenti passa alla linea 5.
5. Visualizza il secondo numero.
6. Vai alla linea 8.
7. Visualizza il primo numero.
8. Fine.

Questa non è la soluzione più ordinata, ma è un primo approccio al problema. Cercheremo delle soluzioni più pulite quando torneremo a trattare i metodi di riordinamento in una delle prossime unità.

Obiettivi del capitolo 2

Ora che avete completato questa unità, verificate di essere in grado di:

Combinare la stampa di caratteri alfanumerici e la stampa di variabili nelle istruzioni PRINT ☐

Usare la , per inserire degli spazi tra istruzioni PRINT ☐
Usare GO TO per ripetere l'uso di un programma ☐
Usare un valore finale per terminare un programma ☐
Usare IF ... THEN GO TO ... ☐
Trovare lo stato logico di asserzioni che comprendono >, <, = ☐
Costruire un diagramma di flusso ☐
Inserire dei contatori nei diagrammi di flusso e nei programmi per controllare l'uso ripetuto di parti di un programma o di un diagramma di flusso ☐

Risposte ai TEST ed agli Esercizi

TEST 1

(a) AREA48
(b) LUNGH.8   LARGH.6
(c) LUNGH.   LARGH.
    8   6

(d) PRINT "LUNGH.",B,
(e) PRINT "LUNGH.",B
PRINT "LARGH.",C
PRINT "AREA",A
(f) PRINT "LUNGH.","LARGH."

TEST 2

Tutto ciò che dovete fare è aggiungere la riga

35 GO TO 5

TEST 3

```
 5 REM **CALCOLO QUADRATI**
10 INPUT N
15 IF N=-9999 THEN GOTO 50
20 LET S=N*N
30 PRINT S
40 GOTO 10
50 STOP
```

Programma 13

(Da notare che questo valore finale non è altrettanto soddisfacente come usare -9999 nel programma sul calcolo dei voti. Infatti voi non potete avere un voto di -9999, ma potreste invece cercare il quadrato di -9999 e questo programma non è in grado di darlo.)

TEST 4

Valori		Asserzione		
A	B	Espressione	Suo valore	Suo stato logico
3	7	A>B	3>7	F
5	3	A>B	5>3	V
—3	5	A>B	—3>5	F
8	5	A<B	8<5	F
3	9	A<B	3<9	V
8	—2	A<B	8<—2	F

A	B
B	A

TEST 5

(a) 40 (b) 100 (c) 40
(d) 40 (e) 40

Il vostro computer può aiutarvi a risolvere il problema. Le istruzioni

```
  40 PRINT "40"
```

e

```
 100 PRINT "100"
```

causano la visualizzazione della linea appropriata.

I seguenti programmi mostrano come potreste avere risolto i primi due problemi (a) (b).

Programma per risolvere (a)

```
  5 REM **PROGRAMMA 14**
 10 LET A=7
 20 LET B=-8
 30 IF A-B<0 THEN GOTO 100
 40 PRINT "40"
 50 GOTO 999
100 PRINT "100"
999 STOP
```

Programma 14

Programma per risolvere (b)

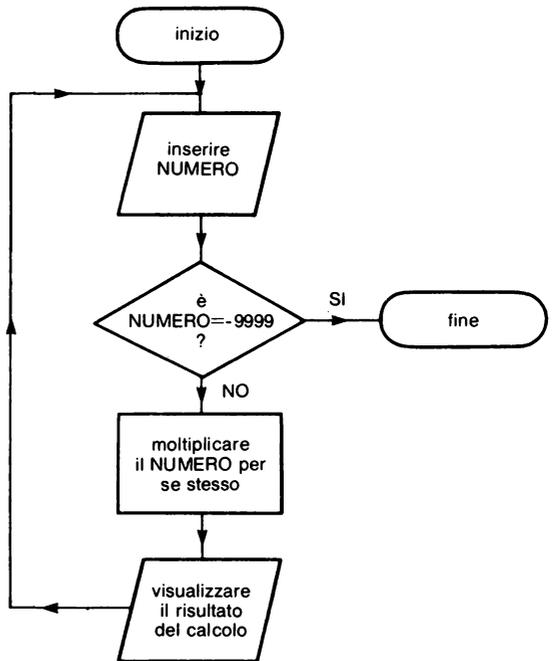
```
  5 REM **PROGRAMMA 15**
 10 LET X=3
 20 LET Y=-3
 30 IF X/Y=-1 THEN GOTO 100
 40 PRINT "40"
 50 GOTO 999
100 PRINT "100"
999 STOP
```

Programma 15

Facendo girare il Programma 14 si ha

```
40
9/999
```

Facendo girare il Programma 15 si ha

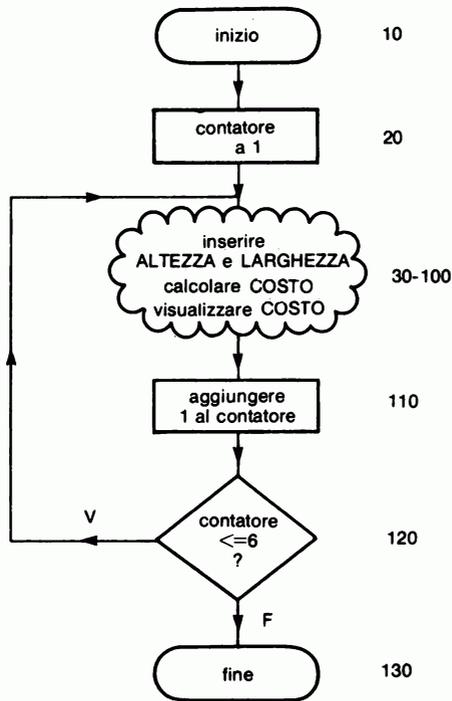
```
100
9/999
```

TEST 8

(a) 10 (b) 16

TEST 9

```
30 IF C=4 THEN GOTO 60
```

Esercizio 1

1(a)

1(b)

```
10 REM **COSTO INSTALLAZIONE
        FINESTRE**
20 LET C=1
30 PRINT "INSERIRE ALTEZZA IN
METRI"
40 INPUT H
50 PRINT H
60 PRINT "INSERIRE LARGHEZZA I
N METRI"
70 INPUT W
```

Il contatore è formato dalle righe 20, 110 e 120

```
  80 PRINT W
  90 LET K=35000*(H+W)+100000*H*
W+125000
 100 PRINT "LA FINESTRA ";C;" CO
STA ";K
 110 LET C=C+1
 120 IF C<=6 THEN GOTO 30
 130 STOP
```

Programma 16

RUN

```
INSERIRE ALTEZZA IN METRI
1.5
INSERIRE LARGHEZZA IN METRI
2
LA FINESTRA 1 COSTA 547500
INSERIRE ALTEZZA IN METRI
1.5
INSERIRE LARGHEZZA IN METRI
3
LA FINESTRA 2 COSTA 732500
INSERIRE ALTEZZA IN METRI
1.5
INSERIRE LARGHEZZA IN METRI
4
LA FINESTRA 3 COSTA 917500
INSERIRE ALTEZZA IN METRI
2.5
INSERIRE LARGHEZZA IN METRI
2
LA FINESTRA 4 COSTA 782500
INSERIRE ALTEZZA IN METRI
1
```

5/40 ———— Messaggio: fine spazio sullo schermo. Premere CONT e NEW LINE.

```
INSERIRE LARGHEZZA IN METRI
2
LA FINESTRA 5 COSTA 430000
INSERIRE ALTEZZA IN METRI
0.5
INSERIRE LARGHEZZA IN METRI
1
LA FINESTRA 6 COSTA 227500
```

1(c)

```
 10 REM **COSTO INSTALLAZIONE
          FINESTRE**
 12 PRINT "INSERIRE NUMERO FINE
STRE ";
 14 INPUT N
 16 PRINT N
 17 PRINT
 20 LET C=1
 30 PRINT "INSERIRE ALTEZZA IN
METRI ";
 40 INPUT H
 50 PRINT H
 60 PRINT "INSERIRE LARGHEZZA I
N METRI ";
 70 INPUT W
 80 PRINT W
 90 LET K=35000*(H+W)+100000*H*
W+125000
100 PRINT "LA FINESTRA ";C;" CO
STA ";K
105 PRINT
110 LET C=C+1
120 IF C<=N THEN GOTO 30
130 STOP
```

Programma 17

RUN

```
INSERIRE NUMERO FINESTRE 3

INSERIRE ALTEZZA IN METRI 1
INSERIRE LARGHEZZA IN METRI 1.5
LA FINESTRA 1 COSTA 362500

INSERIRE ALTEZZA IN METRI 2
INSERIRE LARGHEZZA IN METRI 1
LA FINESTRA 2 COSTA 430000

INSERIRE ALTEZZA IN METRI 2.5
INSERIRE LARGHEZZA IN METRI 1
LA FINESTRA 3 COSTA 497500
```

CAPITOLO 3

STRINGHE

3.1 Cos'è una stringa?

Le prime due unità concernevano l'elaborazione numerica. Il profano spesso vede il computer come un 'masticatore di numeri' ma questa non è certamente la principale funzione di un computer, specialmente in ambiente commerciale. In questa Unità vedremo come il computer può essere usato per manipolare caratteri, usando il linguaggio BASIC.

Per caratteri intendiamo l'alfabeto maiuscolo, le dieci cifre 0-9, i segni di punteggiatura ed alcuni caratteri speciali, esposti qui di seguito:

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, :, ;, >, =, <, ?, Spazio, £, ", (,), +, *, —, /.

Avete imparato a scrivere programmi che usano numeri (3,57,-92,ecc.) e variabili (A,G,Z,ecc.). Il BASIC ci consente anche di inserire nel computer dei gruppi di caratteri.

A questi gruppi di caratteri si fa riferimento come stringhe. Alcuni esempi di stringhe sono:

GATTO	(una parola)
MARGARET THATCHER	(un nome)
Z9)?27	(una mistura di caratteri)
GE 456768	(una targa di auto)

vale a dire che una stringa può essere formata da una qualsiasi combinazione di caratteri - anche uno spazio è un carattere molto importante in una stringa!

Per quanto riguarda il BASIC, un numero è trattato come tale quando viene usato in qualche operazione aritmetica, altrimenti viene considerato come una stringa di caratteri numerici. Quando osserviamo la targa di un'auto o un numero di telefono li vediamo come un gruppo di caratteri numerici, non usiamo questi insiemi di cifre per fare nessun calcolo aritmetico.

Locazioni di memoria per le stringhe

Come possiamo segnalare al computer che il gruppo di caratteri che abbiamo inserito deve essere trattato come numero per usi aritmetici, o solo come una stringa di caratteri? In BASIC la distinzione viene fatta in base alle etichette che diamo alle locazioni di memoria nelle quali mettiamo i caratteri. Se il nome di una locazione di memoria è seguito dal simbolo caratteri che vengono inseriti in tale locazione sono trattati come stringa di caratteri.

Avete visto nell'Unità 1 che le locazioni di memoria per i numeri nel BASIC dello ZX81 sono le 286 locazioni:

A, A0, A1, ... A9
B, B0, ecc.

Le locazioni di memoria per stringhe nel BASIC dello ZX81 sono le 26 locazioni:

A$, B$, C$,...Z$.

Potete leggere le locazioni di memoria per stringhe come:

A$	A stringa	o	A dollaro
B$	B stringa	o	B dollaro

Così ora potete immaginare un microcomputer che abbia due aree per memorizzare i dati: una per i numeri ed una per i caratteri. Questo è illustrato in Figura 1.

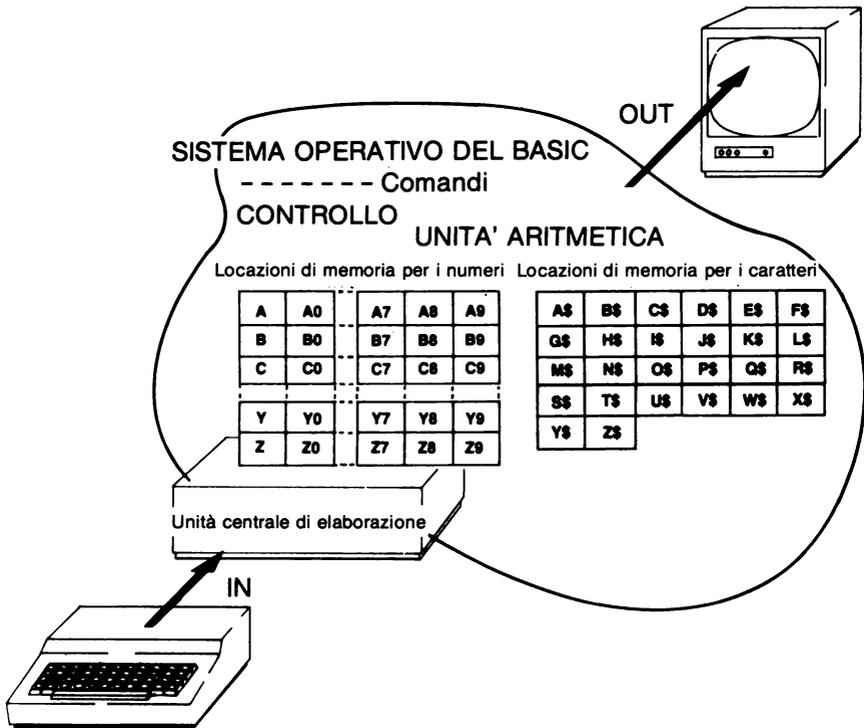



Figura 1 Il nostro sistema come è finora

" " con stringhe

Naturalmente dobbiamo mostrare al computer che vogliamo che la nostra stringa di caratteri sia trattata come tale. Per fare questo mettiamo delle virgolette " " agli estremi della stringa. Per cui scriviamo

```
10 LET Q$="SALVE"
```

e

```
30 IF Q$="SALVE" THEN GOTO 80
```

e così via.

TEST 1

Quali dei seguenti nomi di locazioni di memoria sono validi per stringhe?

(a) A$ (b) M8 (c) T7$ (d) B9 (e) C$3 (f) 8P$ (g) 2$ (h) K$

TEST 2

Quali delle seguenti istruzioni BASIC sono corrette?

(a) `LET A=87`
(b) `LET B$="FRED"`
(c) `LET M$=9583`
(d) `LET K8="JAM POT"`
(e) `LET L17=38`

3.2 Altro sulle stringhe

Una domanda pertinente a questo punto è: 'quanto può essere lunga una stringa?'. In altre parole, quanti caratteri possono essere inseriti, memorizzati o emessi come un singolo gruppo? Lo ZX81 consente alle stringhe di essere di qualsiasi lunghezza entro la capacità di memoria disponibile nel computer. Ma inizialmente in questo corso abbiamo supposto che una locazione di memoria per stringhe possa contenere fino a 40 caratteri, e che questa restrizione si applichi anche nell'inserimento e nell'emissione di stringhe. Così ora dovete pensare ad una locazione di memoria di una stringa, non solo come ad una scatola con una etichetta, ma come ad una locazione con 40 sotto-divisioni, come visibile in Figura 2.

Figura 2 Dimensione delle locazioni di memoria delle stringhe

Abbiamo discusso delle stringhe come se fossero una novità ma voi le avete già incontrate nell'Unità 1. Li abbiamo usato l'istruzione PRINT per visualizzare dei messaggi contenuti tra virgolette. Davamo per implicito che la stringa compresa tra le virgolette venisse visualizzata letteralmente carattere per carattere. Quindi abbiamo visto nell'Unità 2 come delle virgole poste tra gli elementi di una istruzione PRINT introducano degli spazi nella visualizzazione di stringhe.

3.3 PRINT ...;...

Da quanto abbiamo esposto finora potrete rendervi conto che la disposizione delle informazioni sullo schermo è molto importante, sia per le stringhe che per i numeri.

Quando trattiamo informazioni in forma di testo, vale a dire stringhe di caratteri in forma di parole o codici, vogliamo che queste vengano stampate come in una frase e non spaziate attraverso le zone di stampa dello schermo. L'istruzione PRINT ...;... produce tale effetto per noi. PRINT H prenderà i caratteri dalla locazione di memoria H li stamperà sul lato sinistro dello schermo immediatamente seguiti dai caratteri della locazione T$.

Nelle prossime pagine simuleremo un servizio di registrazione di dati del prossimo futuro e lo useremo per dimostrare l'inserimento e l'emissione di stringhe. Iniziamo scrivendo un programma che simuli un servizio di segreteria telefonica.

```
  10 REM *SEGRETERIA TELEFONICA*
  20 PRINT "SALVE"
  30 PRINT "PER CORTESIA SPECIFI
CATE IL VO- STRO NUMERO DI TELEF
ONO"
  40 INPUT T$
  50 PRINT T$
  60 PRINT
  70 PRINT "SALVE",T$
  80 PRINT
  90 PRINT "SALVE";T$
 100 PRINT
 110 PRINT "SALVE ";T$
 120 PRINT
 130 PRINT "SALVE";" ";T$
 140 STOP
```

questa stampa una linea vuota (riga 60)

Programma 1 Stampa di stringhe

Per aiutarvi ad analizzare questo programma abbiamo aggiunto qui sotto una 'traccia' a fianco di un funzionamento tipo. La traccia indica quale linea nel programma produce ciascun risultato.

RUN	Traccia
SALVE	...20
PER CORTESIA SPECIFICATE IL VO- STRO NUMERO DI TELEFONO	...30
58354	...40
	...50
SALVE 58354	...60
	...70
SALVE58354	...80
	...90
SALVÉ 58354	...100
	...110
SALVE 58354	...120
	...130

Commenti

Traccia 40 — INPUT T$ produce " L ". La nostra risposta è stata 58354 che il computer tratta come una stringa e non come un numero. (Se avessimo scritto INPUT T ,allora 58354 sarebbe stato trattato come un numero.)

Traccia 70 — La PRINT ...,... nella riga 70 stampa il 5 del numero telefonico alla 16esima posizione di stampa della linea visualizzata, mentre

Traccia 90 — La PRINT ...;... nella riga 90 stampa il 5 subito dopo la E di SALVE

Nessuno dei due modi e soddisfacente, ma:

Traccia 110
Traccia 130 Le righe 110 e 130 mostrano dei modi alternativi di introdurre gli spazi richiesti, sia stampando Salve ▽ (come nella 110) sia inserendo la stringa ▽ alla sua destra nell'istruzione di uscita (come nella 130). (▽ significa uno spazio.)

K Programma 1.

TEST 3

Studiate questo programma ed indicate come saranno stampate le stringhe. Scrivete il risultato nella griglia qui sotto.

```
 10 PRINT "IMPAGINAZIONE"
 20 LET B$="CORSO DI"
 30 LET C$="BASIC"
 40 PRINT
 50 PRINT B$,C$
 60 PRINT
 70 PRINT B$;C$
 80 PRINT
 90 PRINT B$;" ";C$
100 STOP
```

Programma 2

TEST 4

Scrivete un programma che dovrebbe stampare quanto segue:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
B	A	S	I	C												B	A	S	I	C
B	A	S	I	C	B	A	S	I	C	B	A	S	I	C	B	A	S	I	C	
B	A	S	I	C		B	A	S	I	C		B	A	S	I	C				

3.4 INPUT "...";...

Noi abbiamo usato la PRINT "..." come un segnale di 'pronto' per le istruzioni di input. Molti BASIC hanno la caratteristica di consentirci di combinare queste due istruzioni in una. Per cui nel precedente programma, noi potremmo sostituire:

```
  30 PRINT "PER CORTESIA SPECIFI
CATE IL VO- STRO NUMERO DI TELEF
ONO"
  40 INPUT T$
```

con

```
  40 PRINT "PER CORTESIA SPECIFI
CATE IL VO- STRO NUMERO DI TELEF
ONO";T$
```

Questa prestazione non è disponibile sullo ZX81 ma lo stesso risultato può essere ottenuto usando le righe:

```
  40 PRINT "PER CORTESIA SPECIFI
CATE IL VO- STRO NUMERO DI TELEF
ONO";
  41 INPUT T$
  42 PRINT T$
```

Notate il ';' alla fine della linea 50 che fa in modo che la T$ stampata dopo la frase delle linee 30 e 35.

Il prossimo programma dimostra vari effetti con PRINT. Poiché dobbiamo usare "SALVE" diverse volte per prima cosa lo memorizziamo nella locazione H$ all'inizio del programma.

```
  10 REM *SEGRETERIA TELEFONICA*
  20 LET H$="SALVE"
  30 PRINT H$
  40 PRINT "QUALE E IL VOSTRO TE
LEFONO ? ";
  45 INPUT T$
  50 PRINT T$
  60 PRINT
  70 PRINT H$,T$
  80 PRINT
  90 PRINT H$;T$
 100 PRINT
 110 PRINT H$;" ";T$
 120 STOP
```

Programma 3 Il computer fa la domanda

```
RUN
SALVE
QUALE E IL VOSTRO TELEFONO ? 583
54
SALVE                 58354
SALVE58354
SALVE 58354
```

K Programma 3.

TEST 5

Cosa dovrebbe apparire sullo schermo quando viene eseguito il seguente programma supponendo che il vostro nome sia ENZO ROSSI e la vostra età di 45 anni?

```
   5 REM **TEST 5**
  10 LET T$="GRAZIE"
  20 PRINT "QUALE E IL VOSTRO NO
ME ? "
  30 INPUT N$
  40 PRINT N$
  50 PRINT
  60 PRINT "QUANTI ANNI AVETE ?
"
  70 INPUT A$
  80 PRINT A$
  90 PRINT
 100 PRINT
 110 PRINT T$;"    ";N$;"    ";A$
 120 STOP
```

Programma 4

3.5 Numeri e stringhe in istruzioni PRINT

Avremmo potuto inserire il numero di telefono del precedente programma in una locazione di memoria numerica naturalmente avremmo, dei problemi se il numero fosse troppo lungo, o contenesse degli spazi (per es. 06 735 99 468). Confrontiamo come il BASIC ritornerebbe questo dato da una locazione di memoria numerica o stringa.

In questo programma notate il modo in cui viene usata la stringa di caratteri in S per stampare una scala di riferimento sullo schermo.

```
  10 REM *SEGRETERIA TELEFONICA*
  20 LET H$="SALVE"
  30 LET S$="12345678901234567890
012345"
  35 PRINT H$
  40 PRINT "QUALE E IL VOSTRO TE
LEFONO ? ";
  45 INPUT T$
  50 PRINT T$
  55 PRINT "PER FAVORE SCRIVILO
ANCORA"
  56 INPUT T
  57 PRINT T
  58 PRINT
  60 PRINT S$
  70 PRINT H$,T$
  75 PRINT H$,T
  80 PRINT S$
  90 PRINT H$;T$
  95 PRINT H$;T
 100 PRINT S$
 110 PRINT H$;"  ";T$
 115 PRINT H$;"  ";T
 120 STOP
```

Programma 5 Stampa di stringhe e numeri

RUN	Traccia
SALVE QUALE E IL VOSTRO TELEFONO ? 586 32 PER FAVORE SCRIVILO ANCORA 58632	
1234567890123456789012345	60 S$ numera ciascuna posizione di stampa della pagina.
SALVE 58632 SALVE 58632	75 Notate che la prima cifra 5 è posizionata alla 17a posizione.
1234567890123456789012345 SALVE58632 SALVE58632	95 **La 16a è riservata per il segno del numero in T, ma se il segno è '+' non viene stampato.**
1234567890123456789012345 SALVE 58632 SALVE 58632	115

RUN | **Traccia**

```
SALVE
QUALE E IL VOSTRO TELEFONO ? -96
37
PER FAVORE SCRIVILO ANCORA
-9637

1234567890123456789012345
SALVE               -9637
SALVE               -9637
1234567890123456789012345
SALVE-9637
```

75 notate l'effetto con T$ e che contengono entrambe —9637

```
SALVE-9637
1234567890123456789012345
SALVE -9637
SALVE -9637
```

95

120

K Programma 5

TEST 6

Scrivete un programma per inserire il vostro nome come stringa e la vostra età come numero e visualizzare il messaggio, 'IL MIO NOME È   ED HO   ANNI', riempiendo gli spazi vuoti con le relative informazioni usando una spaziatura normale.

Servizio di registrazione dati

Il seguente è un altro esempio di come si ottengono dei formati di stampa in BASIC. Possiamo immaginare che nel prossimo futuro il nostro televisore, telefono e computer saranno collegati insieme come terminale intelligente. Dopo aver visto una pubblicità interessante noi potremmo comporre un numero telefonico a cui potrebbe seguire il seguente dialogo.

```
SALVE...
QUESTO È UN SERVIZIO DI REGISTRA
ZIONE DATI

PER FAVORE INSERITE LE INFORMA-
ZIONI RICHIESTE

NOME? ENZO ROSSI
TELEFONO? 23685
VIA E NUMERO? MAZZINI 76
CITTA? FIRENZE
CAP ? 50100


GRAZIE PER LE INFORMAZIONI

I VOSTRI DATI SONO STATI REGI-
STRATI COME:



NOME ENZO ROSSI
TELEFONO 23685
INDIRIZZO MAZZINI 76
          50100 FIRENZE

LE SARANNO INVIATE INFORMAZIONI
SUI NOSTRI PRODOTTI ESERVIZI


I VOSTRI DATI PERSONALI
RIMARRANNO RISERVATI
```

Traccie

20
30
40
50
60
70,80
90,100
110,120
130,140
150,160
170,180
190
200
210
240
250
260
266
270
280
290
300
320
330
336
340
350

(Naturalmente, 'Le saranno inviate' potrà eventualmente essere anche 'saranno visualizzate ora sul vostro terminale', ed allora la sola informazione che sarà necessario inserire potrebbe essere un codice di abbonamento.)

Il dialogo simulato viene ottenuto con il seguente programma:

```
  10 REM **QUESTO E UN SERVIZIO
          DI REGISTRAZIONE
          DATI**
  30 PRINT "SALVE..."
  40 PRINT "QUESTO E UN SERVIZIO
 DI REGISTRAZIONE DATI"
  50 PRINT
  60 PRINT "PER FAVORE INSERITE
LE INFORMA- ZIONI RICHIESTE"
  70 PRINT
  80 PRINT "NOME? ";
  81 INPUT N$
  82 PRINT N$
  90 PRINT "TELEFONO? ";
  91 INPUT T$
  92 PRINT T$
 110 PRINT "VIA E NUMERO? ";
 111 INPUT R$
 112 PRINT R$
 120 PRINT "CITTA? ";
 121 INPUT C$
 122 PRINT C$
 130 PRINT "CAP ? ";
 131 INPUT P$
 132 PRINT P$
 140 PRINT
 150 PRINT
 160 PRINT
 170 PRINT "GRAZIE PER LE INFORM
AZIONI"
 180 PRINT
 190 PRINT "I VOSTRI DATI SONO S
TATI REGI-  STRATI COME:"
 195 SCROLL
 196 SCROLL
 200 PRINT "NOME ";N$;" ";
 201 SCROLL
 204 PRINT "TELEFONO ";T$
 205 SCROLL
 210 PRINT "INDIRIZZO ";R$
 215 SCROLL
 220 PRINT "              ";P$;" ";
C$
 230 SCROLL
 235 SCROLL
 240 PRINT "LE SARANNO INVIATE I
NFORMAZIONI"
 241 SCROLL
 242 PRINT "SUI NOSTRI PRODOTTI
ESERVIZI"
```

```
245 SCROLL
250 SCROLL
255 SCROLL
260 PRINT "I VOSTRI DATI PERSON
ALI"
265 SCROLL
270 PRINT "RIMARRANNO RISERVATI
"
275 STOP
```

Programma 6 Servizio registrazione dati

K Programma 6.

3.6 Lettere standard

Un servizio di registrazione dati come quello che abbiamo appena visto potrà esserci in futuro, ma le lettere standard personalizzate sono già oggi una realtà. Una lettera del genere dovrebbe essere composta su un word-processor (un programma per elaborare testi), ma se il vostro microcomputer non ha disponibili delle capacità di word processor, voi potete ottenere dei risultati più limitati usando il BASIC.

Esempio 1

L'ufficio assunzione personale di una banca riceve molte richieste di impiego. La sua politica è di intervistare inizialmente i possibili candidati alla assunzione presso la filiale locale. Una lettera standard viene inviata dall'ufficio assunzione a ciascun candidato con i dettagli individuali della intervista proposta. Sviluppiamo un programma BASIC per scrivere tale lettera.

Soluzione

Il seguente programma dovrebbe fare quanto richiesto.

Per la spiegazione del significato delle stringhe di INPUT da A$ fare riferimento alle istruzioni REM nel programma.

```
10 REM *SCRITTORE DI LETTERE*
15 REM **NOME DEL CANDIDATO**
20 INPUT A$
30 REM **DATA LETTERA
        CANDIDATO**
```

```
  35 INPUT B$
  40 REM **NOME INTERVISTATORE**
  45 INPUT C$
  50 REM **ORA INTERVISTA**
  52 INPUT D$
  55 REM **GIORNO INTERVISTA**
  60 INPUT E$
  65 REM **LUOGO INTERVISTA**
  70 INPUT F$
  75 REM **NOME IMPIEGATO CHE
          RISPONDE**
  80 INPUT G$
  85 REM **FINE INSERIMENTO
          DATI**
 100 REM **STAMPA LETTERA**
 110 PRINT
 120 PRINT
 140 PRINT "EGR. SIG.";A$;","
 150 PRINT
 160 PRINT "GRAZIE PER LA VS. LE
TERA DEL"
 170 PRINT B$;".";" LÁ INVITIAMO
A"
 180 PRINT "RECARSI PER UN COLLO
QUIO CON IL"
 190 PRINT "SIG. ";C$;" ALLE ";D
$;" DEL"
 200 PRINT E$;" PRESSO IL NOSTRO
"
 210 PRINT "UFFICIO DELLA FILIAL
E DI ";F$;"."
 220 PRINT
 230 PRINT "CORDIÀLI SALUTI,"
 240 PRINT
 250 PRINT
 260 PRINT
 270 PRINT G$
 280 STOP
```

Programma 7 Lettera di convocazione di una banca

Tutto questo dovrebbe avere per risultato il seguente funzionamento:

RUN	Traccia
ROSSI	20
16 NOVEMBRE	35
BIANCHI	45
16.30	52
20 NOVEMBRE	60
VERONA	70
FRANCO SAVINI	80

```
EGR. SIG.ROSSI,                                         140
                                                        150
GRAZIE PER LA VS. LETERA DEL                            160
16 NOVEMBRE. LA INVITIAMO A                             170
RECARSI PER UN COLLOQUIO CON IL                         180
SIG. BIANCHI ALLE 16.30 DEL                             190
20 NOVEMBRE PRESSO IL NOSTRO                            200
UFFICIO DELLA FILIALE DI VERONA.                        210
                                                        220
CORDIALI SALUTI,                                        230
                                                        240
                                                        250
                                                        260
FRANCO SAVINI                                           270
```

L'utilizzatore del programma potrebbe trovarlo difficile da usare, poiché tutto ciò che ottiene è una serie di "[L]", potrebbe, realizzare un semplice appunto per ricordarsi della struttura della lettera.

```
A$
B$
C$
D$
E$
F$
G$

R. A$

GRAZIE PER LA VS. LETTERA DEL B$.
LA INVITIAMO A RECARSI PER UN COLLOQUIO CON IL
C$ E D$ SU E$
NEL NOSTRO UFFICIO
DELLA FILIALE DI F$.

CORDIALI SALUTI,

G$
```

* *Programma 7.*

K

Esercizio 1

Un agente immobiliare invia periodicamente una lettera per controllare se i suoi clienti sono ancora alla ricerca di una proprietà, e che i dettagli della loro richiesta (per es. il tipo di proprietà, il prezzo, ecc.) sono corretti. Realizzate un programma in BASIC per scrivere tale lettera.

Esercizio 2

Noi tutti scriviamo lettere per richiedere qualcosa, per es. dettagli di un prodotto, un servizio, un lavoro, ecc. Realizzate un programma in BASIC per scrivere una lettera che copra la gamma più ampia possibile di applicazioni, lasciando da inserire solo i dettagli specifici di ciascuna richiesta.

3.7 Files e Records

Abbastanza spesso noi dobbiamo conservare dei dati della stessa specie, vale a dire creare un 'file' di informazioni. Una rubrica telefonica è un buon esempio di cosa nell'elaborazione dati viene chiamato un 'file', che una collezione di 'records' simili. Ogni record ha la forma:

ed è formato da un certo numero di campi - in questo caso tre: nome, indirizzo e telefono. Un record, quindi, è un insieme di campi ed un file è un insieme di records. Una rubrica telefonica è organizzata in ordine alfabetico di nomi il che gli conferisce una struttura semplice.

Comparazione di stringhe

Possiamo voler comparare delle stringhe. Supponiamo, per esempio, di avere nel nostro microcomputer una rubrica telefonica personale e che vogliamo verificare se ROSSI è nella nostra lista. Il computer dovrà confrontare la stringa "ROSSI" con tutte le stringhe nel campo dei nomi della nostra rubrica. Ciò può essere fatto molto facilmente perché ciascuna lettera è rappresentata all'interno del computer da un codice binario. Per cui

A è 100 0001
B è 100 0010

e così via. (Vedere l'Appendice a questa Unità per una lista completa dei codici.) Così parole sistemate in ordine alfabetico sulla carta saranno rappresentate nel computer da codici in ordine numerico.

Per cui se

A$= CAT
B$= DOG
C$= CAT
D$= FISH
E$= CATS

A$= C$

Ma B$ > A$ (è più avanti nell'alfabeto)
e E$ > A$ (la S in più di cats la colloca dopo CAT nell'ordine alfabetico.)

Ora noi useremo questa prestazione nei nostri esempi.

Esempio 2

Creiamo un file dati di nomi e dei relativi numeri telefonici. Scriviamo un programma in BASIC per cercare attraverso il file e trovare un particolare nome, e se viene trovato visualizzare il numero di telefono ad esso associato.

Soluzione

Noi possiamo tentare un algoritmo descrittivo come il seguente:

1. Inizio.
2. Inserire il nome da cercare.
3. Leggere il prossimo record del file dati (vale a dire nome e numero).
4. Se la fine del file è stata raggiunta allora visualizzare il messaggio 'non trovato nel file' e vai alla riga 7 altrimenti continua con la riga 5.
5. Se il nome cercato = nome dato allora visualizzare nome e numero e poi saltare alla riga 7 altrimenti continua con la riga 6.
6. Ritornare alla riga 3 per il prossimo record.
7. Fine.

Però, il BASIC in genere non consente istruzioni complicate come 4 e 5, e così noi abbiamo diviso queste istruzioni come visibile nel prossimo algoritmo:

1. Inizio.
2. Inserire il nome da cercare.
3. Leggere il prossimo record dal file dati.
4. Se la fine del file è stata raggiunta allora vai alla riga 7 altrimenti continua con la riga 5.
5. Se il nome cercato = nome dato allora vai alla riga 9 altrimenti continua con la riga 6.
6. Ritornare alla riga 3 per il prossimo record.
7. Visualizzare il messaggio 'non è nel file'.
8. Fine.
9. Visualizzare nome e numero.
10. Fine.

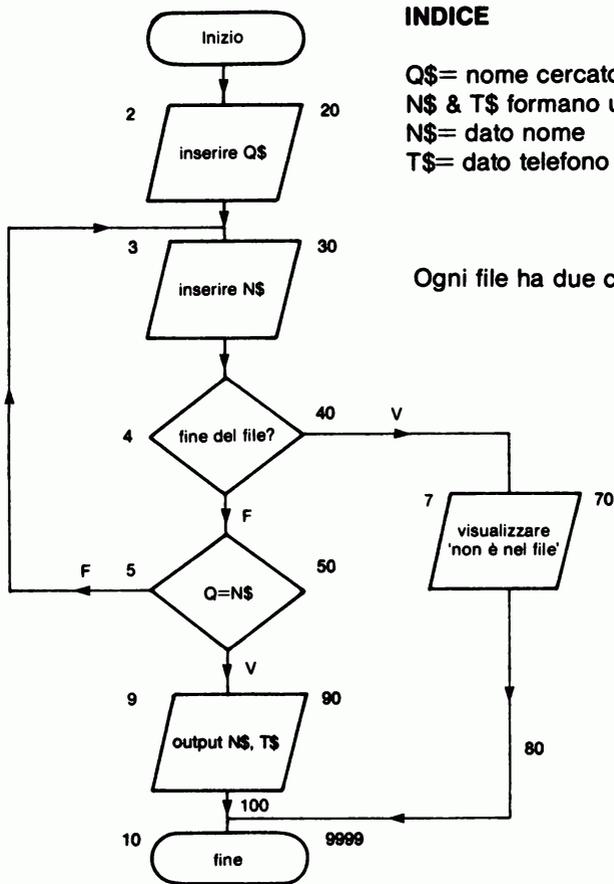



Figura 3 Ricerca in un archivio telefonico

Ora ogni record contiene due campi: nome e numero.

per es.

Campo 1	Campo 2
Nome N$	Telefono T$
LAURA	49456

così ciascuna voce deve contenere le informazioni per ciascun campo. A questo punto del corso è sufficiente usare solo alcuni dati come esempio. Il primo paio di dati

LAURA 1234

viene memorizzato come

```
30 LET N$="LAURA"
32 LET T$="1234"
```

ed il secondo paio come

```
40 LET N$="MARIO"
42 LET T$="9823"
```

ecc. Questo è un po' complicato ma ci consente di cercare nella lista per trovare la linea nella quale Q$=N$.

```
10 REM **RUBRICA TELEFONICA**
20 PRINT "NOME DELLA PERSONA C
ERCATA?"
21 INPUT Q$
22 PRINT Q$
25 REM **DATI**
30 LET N$="LAURA"
32 LET T$="1234"
35 IF Q$=N$ THEN GOTO 200
40 LET N$="MARIO"
42 LET T$="9823"
45 IF Q$=N$ THEN GOTO 200
50 LET N$="PAOLA"
52 LET T$="1850"
```

```
 55 IF Q$=N$ THEN GOTO 200
 190 PRINT Q$;" NON E NELLA LIST
A"
 195 STOP
 200 PRINT "IL NUMERO DI ";Q$;"
E ";T$
 210 STOP
```

Programma 8 Rubrica telefonica

```
NOME DELLA PERSONA CERCATA?
LAURA
IL NUMERO DI LAURA E 1234


NOME DELLA PERSONA CERCATA?
FRANCO
FRANCO NON E NELLA LISTA
```

K Programma 8.

TEST 7

Quali cambiamenti dovreste fare al Programma 8 per inserire il numero di telefono di una persona e ricevere il nome dell'abbonato, o la scritta 'non è nella lista'?

3.8 Riordinamento

Avrete notato che i dati nell'archivio della rubrica telefonica dell'esempio 2 erano in ordine alfabetico come vi sareste aspettati sarebbe difficile un uso normale se così non fosse. Tuttavia, la nostra soluzione a questo problema di ricerca dati non usa questa informazione; noi cerchiamo soltanto attraverso il file di dati, record per record finché troviamo il nome, o raggiungiamo la fine del file. Il nostro algoritmo avrebbe lavorato ugualmente bene anche se i dati non fossero stati in ordine alfabetico. Spenderemo del tempo più avanti nel corso, nell'esame del riordino e della ricerca di dati, in modo che vi rendiate conto dei vantaggi del riordinare i dati in ordine alfabetico.

Iniziamo con questo problema.

Esempio 3

Scriviamo un programma BASIC per inserire due nomi nel computer e visualizzare il nome che dovrebbe venire per primo in ordine alfabetico.

Soluzione

Algoritmo descrittivo

1. Inizio.
2. Inserire il primo nome.
3. Inserire il secondo nome.
4. Se primo nome <secondo nome allora vai alla riga 7 altrimenti continua con la riga 5.
5. Visualizzare il secondo nome.
6. Vai alla riga 8.
7. Visualizzare il primo nome.
8. Fine.

Un diagramma di flusso per la soluzione di questo problema è visibile in Figura 4.

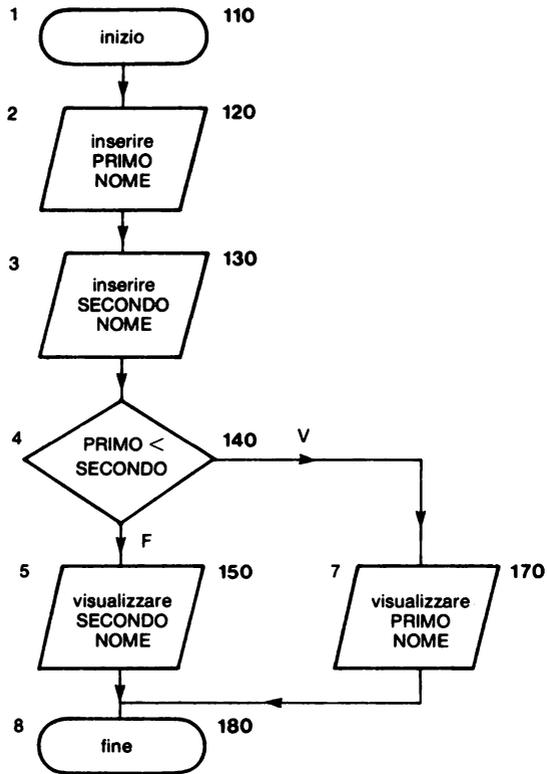



Figura 4 Ricerca tra due nomi del primo in ordine alfabetico

```
110 REM **PRIMO IN ORDINE
          ALFABETICO**
120 PRINT "PRIMO NOME? ";
121 INPUT A$
122 PRINT A$
130 PRINT "SECONDO NOME?";
131 INPUT B$
132 PRINT B$
140 IF A$<B$ THEN GOTO 170
150 PRINT "IL PRIMO IN ORDINE A
LFABETICO"
155 PRINT "E ";B$
160 GOTO 180
170 PRINT "IL PRIMO IN ORDINE A
LFABETICO"
175 PRINT "E ";A$
180 STOP
```

Programma 9

Funzionamenti tipo

```
RUN
PRIMO NOME? BRUNO
SECONDO NOME?LAURA
IL PRIMO IN ORDINE ALFABETICO
E BRUNO

RUN
PRIMO NOME? LAURA
SECONDO NOME?BRUNO
IL PRIMO IN ORDINE ALFABETICO
E BRUNO
```

K Programma 9.

Il gioco delle tre carte

Supponiamo ora di volere inserire tre nomi e visualizzare il nome che viene per primo in ordine alfabetico. Una soluzione standard a questo problema potrebbe essere quella di Figura 5.

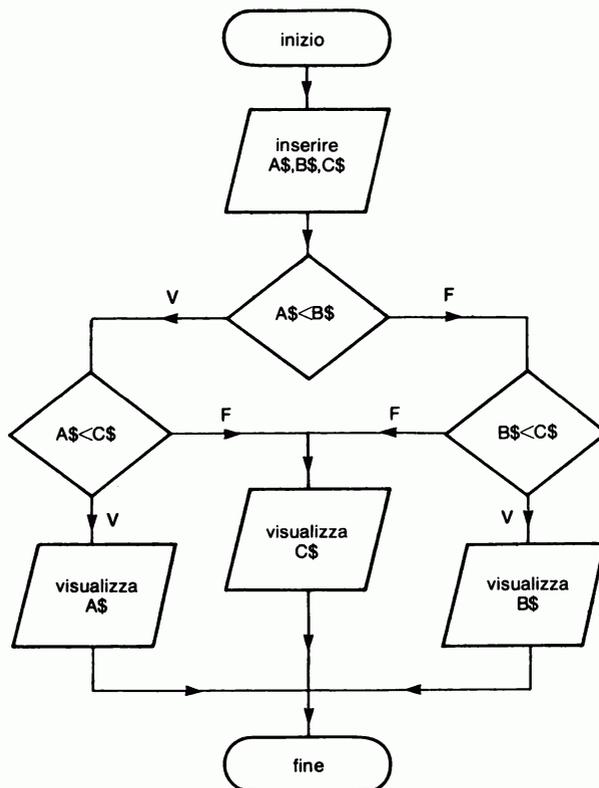



Figura 5 Ricerca fra tre nomi del primo in ordine alfabetico

Quando gli viene chiesto di risolvere questo problema la maggior parte degli studenti presenta una risposta simile all'algoritmo di Figura 5. È una buona soluzione, ma presenta delle difficoltà, derivanti dal fatto che abbiamo a disposizione 3 decisioni e 3 funzioni di uscita. Questo metodo potrebbe richiederci una pazienza ed una abilità eccessive se provassimo a ripeterlo per 4, 5...10 nomi.

Un semplice metodo per risolvere questo problema è l'inserimento dei nomi uno per uno e conservare in A più basso fino a quel momento. Il programma conserva solo il più basso, distruggendo tutti gli altri dati scartati. Nel prossimo esercizio vi suggeriamo di provare questo sistema per risolvere il problema.

Esercizio 3

Scrivete un programma BASIC per inserire tre nomi e visualizzare il nome che dovrebbe venire per primo in ordine alfabetico usando il metodo discusso nei paragrafi precedenti.

Obiettivi del capitolo 3

Ora che avete completato questa unità, verificate di essere in grado di usare quanto segue in semplici programmi:

- Locazioni di memoria per stringhe ☐
- PRINT ...;... ☐
- INPUT "...";... ☐
- IF A$= B$ THEN ... ☐
- Semplici procedure di riordinamento ☐

Risposte ai TEST ed agli Esercizi

TEST 1

Locazioni di stringa valide: A$. K$ e gli altri, M8 e B9 sono locazioni di memoria numeriche.
Il resto dei nomi non è ne di locazioni numeriche ne di stringa.

TEST 2

(a), (b) ed (e) sono corretti sebbene (e) non sia accettabile nel BASIC ridotto.
(c) è errato; M$ è una locazione di stringa per cui deve seguire "9583".
(d) è errato; K8 è una locazione per numeri ad essa non può essere assegnata la stringa "COMPUTER".

TEST 3

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
I	M	P	A	G	I	N	A	Z	I	O	N	E									
C	O	R	S	O							D	I					B	A	S	I	C
C	O	R	S	O	D	I	B	A	S	I	C										
C	O	R	S	O		D	I		B	A	S	I	C								

(Risposta mostrata con le zone di stampa standard.)

TEST 4

```
 5 REM **TEST 4**
10 LET B$="BASIC"
20 PRINT B$,B$
30 PRINT B$;B$;B$;B$
40 PRINT B$;" ";B$;" ";B$
50 STOP
```

Programma 10

TEST 5

QUAL'È IL VOSTRO NOME? ENZO ROSSI
QUAL'È LA VOSTRA ETA'? 45

GRAZIE ENZO ROSSI 45

TEST 6

```
   5 REM **TEST 6**
  10 PRINT "QUALE E IL VOSTRO NO
ME?"
  20 INPUT N$
  30 PRINT N$
  40 PRINT "QUALE E LA VOSTRA ET
A?"
```

```
 50 INPUT A
 60 PRINT A
 70 PRINT "IL MIO NOME E ";N$;"
ED HO"
 80 PRINT A;" ANNI."
```

Programma 11

Esercizio 1

Esercizio 1 e 2 sono molto simili come natura e la risposta all'Esercizio 1 non viene qui inclusa.

Esercizio 2

```
  10 REM *LETTERA DI RICHIESTA*
  40 PRINT "DETTAGLI DELL INDIRI
ZZO"
  50 PRINT
  60 PRINT "NOME...? ";
  65 INPUT N$
  70 PRINT N$
  80 PRINT "VIA...? ";
  85 INPUT S$
  90 PRINT S$
 100 PRINT "CITTA...? ";
 105 INPUT T$
 110 PRINT T$
 120 PRINT
 130 PRINT "DATA DI QUESTA LETTE
RA? ";
 135 INPUT D$
 140 PRINT D$
 150 PRINT
 160 PRINT "DETTAGLI DEL PRODOTT
O/SERVIZIO"
 170 PRINT
 180 PRINT "ARTICOLO CHE INTERES
SA? "
 185 INPUT I$
 190 PRINT I$
 200 PRINT "FONTE DELLA INFORMAZ
IONE?"
 205 INPUT A$
 210 PRINT A$
 220 PRINT "DATA DELLA FONTE? "
 225 INPUT E$
 230 PRINT E$
 240 PAUSE 200
```

Fa aspettare il computer per 4 secondi.
Quindi cancella lo schermo.

```
250 CLS
270 PRINT N$
280 PRINT S$
290 PRINT T$
300 PRINT
310 PRINT D$
320 PRINT
330 PRINT "EGR. SIG.,"
340 PRINT
350 PRINT "VOGLIATE CORTESEMENT
E INVIARMI"
360 PRINT "DETTAGLI SU ";I$;","
370 PRINT "COME DESCRITTO NEL S
EGUENTE:"
380 PRINT A$
390 PRINT "DATATO ";E$;","
400 PRINT
410 PRINT "CORDIALI SALUTI,"
420 PRINT
430 PRINT
440 PRINT "FRANCO SALVI"
```

Programma 12

Funzionamento tipo

```
RUN
DETTAGLI DELL INDIRIZZO

NOME...? MICRO BYTE COMPUTERS S.
R.L.
VIA...? LARGO TOSCANINI, 40
CITTA...? 43100 PARMA

DATA DI QUESTA LETTERA? 12 FEBBR
AIO 84

DETTAGLI DEL PRODOTTO/SERVIZIO

ARTICOLO CHE INTERESSA?
PROGRAMMA WORD PROCESSOR
FONTE DELLA INFORMAZIONE?
COMPUTEWORLD ITALIA
DATA DELLA FONTE?
24 GENNAIO 84
```

```
MICRO BYTE COMPUTERS S.R.L.
LARGO TOSCANINI, 40
43100 PARMA

12 FEBBRAIO 84



EGR. SIG.,

VOGLIATE CORTESEMENTE INVIARMI
DETTAGLI SU PROGRAMMA WORD PROCE
SSOR,
COME DESCRITTO NEL SEGUENTE:
COMPUTERWORLD ITALIA
DATATO 24 GENNAIO 84,

CORDIALI SALUTI,


FRANCO SALVI
```

TEST 7

Modificare le seguenti righe:

```
  20 PRINT "NUMERO DELLA PERSONA
 CERCATA"
 190 PRINT "IL NUMERO DELL ABBON
ATO ";Q;" NON E NELLA LISTA"
 200 PRINT "IL NUMERO DELL ABBON
ATO ";N;" E ";Q
```

Esercizio 3

Un semplice metodo per risolvere questo problema è mostrato nel Programma 13. I nomi sono inseriti uno per uno, ed il più basso fino a quel momento, rimane memorizzato in A$ il programma, tuttavia, conserva soltanto questa informazione; tutti gli altri dati vengono perduti.

```
  10 REM **PRIMO IN ORDINE
          ALFABETICO **
  15 PRINT "ORDINE ALFABETICO"
  16 PRINT
  20 PRINT "PRIMO NOME? ";
  21 INPUT A$
  22 PRINT A$
  30 PRINT "PROSSIMO NOME? ";
  31 INPUT B$
  32 PRINT B$
  40 IF B$="ZZZZ" THEN GOTO 100
  50 IF A$<B$ THEN GOTO 70
  60 LET A$=B$
  70 PRINT "PRIMO FINORA E ";A$
  80 GOTO 30
 100 PRINT A$;" E IL PRIMO DI TU
TTI"
 110 STOP
```

Programma 12

RUN

```
ORDINE ALFABETICO

PRIMO NOME? TOM
PROSSIMO NOME? SID
PRIMO FINORA E SID
PROSSIMO NOME? JOE
PRIMO FINORA E JOE
PROSSIMO NOME? PETE
PRIMO FINORA E JOE
PROSSIMO NOME? FRED
PRIMO FINORA E FRED
PROSSIMO NOME? BILL
PRIMO FINORA E BILL
PROSSIMO NOME? RON
PRIMO FINORA E BILL
PROSSIMO NOME? ALAN
PRIMO FINORA E ALAN
PROSSIMO NOME? ZZZZ
ALAN E IL PRIMO DI TUTTI
```

Appendice

Codice ASCII (American Standard Code for Information Interchange) - (estratti).
(I codici dello ZX81 - dove disponibili - sono indicati tra parentesi).

	64	—	0	48	(28)
A	65	(38)	1	49	(29)
B	66	(39)	2	50	(30)
C	67	(40)	3	51	(31)
D	68	(41)	4	52	(32)
E	69	(42)	5	53	(33)
F	70	(43)	6	54	(34)
G	71	(44)	7	55	(35)
H	72	(45)	8	56	(36)
I	73	(46)	9	57	(37)
J	74	(47)	:	58	(14)
K	75	(48)	;	59	(25)
L	76	(49)	<	60	(19)
M	77	(50)	=	61	(20)
N	78	(51)	>	62	(18)
O	79	(52)	?	63	(15)
P	80	(53)	Spazio	32	(0)
Q	81	(54)	!	33	—
R	82	(55)	”	34	(11)
S	83	(56)		35	
T	84	(57)	$	36	(13)
U	85	(58)	%	37	—
V	86	(59)	&	38	—
W	87	(60)	’	39	—
X	88	(61)	(	40	(16)
Y	89	(62)	)	41	(17)
Z	90	(63)	★	42	(23)
[	91	—	+	43	(21)
\	92	(65)	’	44	(26)
]	93	—	—	45	(22)
↑	94	—	.	46	(27)
—	95	—	/	47	(24)

Commento 1

7 circuiti elettrici ciascuno nello stato di on (1) o off (0) possono essere usati per rappresentare i caratteri mostrati.

CAPITOLO 4

LISTE

4.1 Variabili

Abbiamo già visto come una locazione di memoria possa conservare molti valori diversi durante l'esecuzione di un programma. Perciò il valore in una locazione di memoria può variare durante il funzionamento, e così spesso facciamo riferimento alle locazioni come a variabili in un programma. Perciò le 286 etichette di locazioni:

A, A0, A1, ... A9, B, B0, ... Z8, Z9

sono chiamate variabili numeriche, e le altre 26 locazioni:

A$, B$,...Z$

sono chiamate variabili stringa. Le etichette delle locazioni di memoria sono usate in espressioni nelle istruzioni dei programmi esattamente come i matematici usano le variabili nelle equazioni.

4.2 Liste

Le liste e i supermarket appaiono legati inestricabilmente. Noi vi entriamo con una lista di cose che vogliamo comprare, e ne usciamo con gli articoli e una lista di prezzi sotto forma di scontrino. La lista di prezzi risulta dal processo di trasferimento degli articoli dal carrello alla cassa. Questo è un processo abbastanza casuale ma noi potremmo aver dato alla lista un ordine più significativo in molti modi. Con un po' di lavoro potremmo aver preso gli articoli dal carrello in ordine di prezzo, vale a dire per primo il più economico tutti gli altri e così via con il più costoso preso per ultimo, così che lo scontrino con i prezzi fosse in ordine di costo. Allo stesso modo, avremmo potuto prendere gli articoli dal carrello in ordine di peso, il più leggero per primo ed il più pesante per ultimo; dando così un ordine completamente diverso allo scontrino. La possibilità di mettere in relazione la posizione nella lista con il valore dell'articolo in vari modi è una delle caratteristiche più utili delle liste, come vedremo entro la fine di questa unità.

4.3 Lista di variabili

Molti dei dati che abbiamo considerato finora possono essere classificati in un insieme. Noi abbiamo considerato insiemi di voti, di nomi e numeri telefonici ad essi associati, di nazioni e loro capitali. Molti dati possono essere classificati in vari modi. Se raccogliamo dei dati per qualche scopo, questo stesso scopo offre un insieme di valori con caratteristiche comuni. Ci sono degli evidenti vantaggi a dare un nome alle locazioni di memoria per le voci di un insieme di dati in un modo che

evidenzi che tutte le voci appartengono ad un insieme. Ancora meglio, sarebbe utile se i nomi delle locazioni di memoria identificassero la posizione di una voce all'interno dell'insieme.

Per esempio con una numerazione delle variabili; questo evidenzia che i valori sono in qualche modo associati tra loro, ed assegna loro una posizione all'interno dell'insieme. Questo è ottenuto nel modo seguente.

Considerate un insieme di voti nel registro di un insegnante. Essi formano per natura una lista e possono essere collocati in locazioni di memoria nel modo seguente:

Voce	Simbolo locazione
Il primo nella M-lista è 42	M(1)=42
Il secondo nella M-lista è 67	M(2)=67
Il terzo nella M-lista è 90	M(3)=90
ecc...	

M(1), M(2), M(3) sono come locazioni di memoria separate. Potete prendere una qualsiasi delle 26 locazioni di memoria e aggiungere dei numeri tra parentesi dopo di esse per realizzare delle liste di locazioni di memoria, per es.

Nome lista	Locazioni in questa lista
M(1)	M(1), M(2), M(3) ...
C$(1)	C$(1), C$(2), C$(3) ...

Il numero tra parentesi (qui indicato con 1) è chiamato indice della lista, e può essere un qualsiasi numero intero positivo che rientri nella capacità di memoria del vostro computer.

Liste di stringhe

Come potete vedere dalla tabella qui sopra, le liste possono essere di stringhe come numeriche. Se una lista ha come nome M(1), è chiaramente una lista di numeri, ma M$(1) dovrebbe essere una lista di stringhe, per es. una lista di nomi potrebbe essere memorizzata come:

Indice	Voce	Nome variabile
1 2 3 ecc.	Rossi Franchi Salvi ecc.	N$(1) N$(2) N$(3) ecc.

Figura 1 Lista nomi di stringa

Liste (vettori) e matrici

Ad una tabella di dati, come quella visibile in Figura 1, viene spesso fatto riferimento come vettore di dati. Con i dati visualizzati in righe e colonne in questo modo, ad una tabella viene spesso fatto riferimento come ad un vettore bi-dimensionale. Vedremo come il BASIC provvede ai vettori bi-dimensionali in una prossima unità.

DIM, o quanto è lunga una lista?

È lunga quanto volete; possiamo far si che una lista sia di qualsiasi lunghezza desiderata, provvedendo di avvertire il sistema come prima cosa. Noi facciamo questo con una istruzione DIM che appare nel programma prima che si faccia riferimento al vettore.

DIM per vettori numerici

Se volete usare un vettore, diciamo M, per memorizzare 8 numeri allora annunciate al computer la vostra intenzione con l'istruzione:

DIM M(8)

Questo dice al computer di riservare spazio in memoria per le 8 variabili M(1), M(2), M(3), ..., M(8).

DIM per vettori stringa

Quando volete riservare dello spazio per un vettore di stringhe dovete dire al computer due cose: (a) il numero di stringhe che desiderate memorizzare nel vettore e (b) la loro massima lunghezza. Voi fate ciò con una istruzione DIM della seguente forma:

Questo crea 10 locazioni di memoria N$(1), N$(2), N$(3), ..., N$(10) ognuna capace di contenere una stringa di 15 caratteri.

Voci e numero indice

Il seguente esercizio da fare con carta e matita dovrebbe rinforzare la vostra comprensione di cosa viene indicato dai termini voce ed indice, e la loro spesso momentanea relazione. Esso prepara anche il terreno per la procedura di interscambio-riordinamento che considereremo in dettaglio più avanti in questa unità.

Esempio 1

Trasferiamo la voce di valore più basso nella seguente lista alla posizione 1, confrontando a turno ciascuno dei valori nel resto della lista con il valore presente alla posizione 1. Interscambiate le voci se una delle voci nel resto della lista è più bassa di quella presente nella posizione 1. (È più facile farlo che descriverlo!)

Lista: **3, 42, -8, 9, -11**

Inizio		Stadi di comparazione & interscambio			
posizione o indice	*voce*	*1mo giro c*	*2ndo giro c&i*	*3rzo giro c*	*4rto giro c&i*
1	3	3	-8	-8	-11
2	42	42	42	42	42
3	-8	-8	3	3	3
4	9	9	9	9	9
5	-11	-11	-11	-11	-8

Figura 2 Riordino per sistemare per primo il numero più basso

TEST 1

Riportate la procedura vista nell'Esempio 1 per la seguente lista di numeri:

6, 8, 4, 7, 3, 9, 1.

4.4 Inserimento e stampa liste

Prima che possiamo manipolare le voci di una lista dobbiamo inserire la lista di voci nel computer, e di solito anche stampare la stessa lista dopo l'elaborazione.

Esempio 2

Scriviamo un programma per inserire tre numeri in una lista e stampare gli elementi della lista in ordine inverso.

Soluzione

Possiamo chiamare la lista A(I). Il programma richiesto è allora il seguente:

```
10 REM **ESEMPIO 2**
30 DIM A(3)
40 INPUT A(1)
50 INPUT A(2)
60 INPUT A(3)
```

```
70 PRINT
80 PRINT
90 PRINT A(3),A(2),A(1)
```

Programma 1 Inversione dell'ordine della lista

Funzionamento tipo

```
RUN
29
32
-17

-17                 32
29
```

K Programma 1.

Abbiamo fatto quanto richiesto nel problema, ma non abbiamo fatto nessun progresso significativo nella tecnica di programmazione poiché avremmo potuto fare lo stesso lavoro con le tecniche descritte nella prima unità semplicemente chiamando le variabili P, Q ed R e stampandole come R, Q, P. Per far meglio il lavoro abbiamo bisogno di contare la lista come viene inserita così che noi possiamo usare il contatore in ordine inverso quando stampiamo la lista. Il prossimo esempio aggiunge questa rifinitura.

Contare una lista

Esempio 3

Scriviamo un programma per inserire cinque numeri in una lista, per visualizzarne le voci della lista ed il loro indice in forma di tabella, e quindi stampare gli elementi della lista in ordine inverso.

Soluzione

Come elencato nella domanda, ci sono tre parti principali nella soluzione e noi possiamo visualizzarle in forma di diagramma di flusso come in Figura 3.

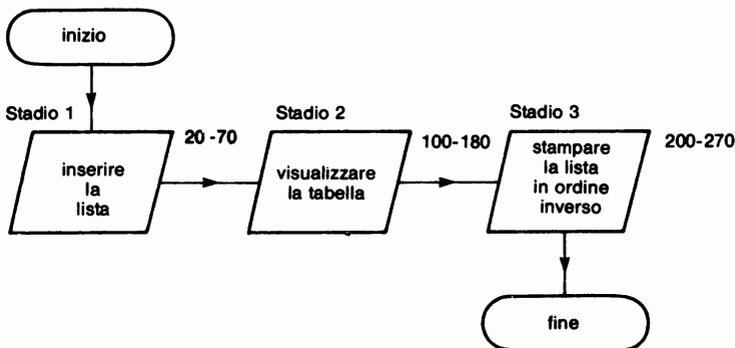



Figura 3 Stadi nella soluzione dell'Esempio 3

Stadio 1 Noi usiamo la variabile C per contare gli elementi della lista inserita, e per agire come indice per la L-lista.

```
 10 REM **INSERIRE UNA LISTA**
 20 DIM L(5)
 30 LET C=1
 40 PRINT "INSERIRE IL PROSSIMO
NUMERO"
 45 INPUT L(C)
 50 PRINT L(C)
 60 LET C=C+1
 70 IF C<=5 THEN GOTO 40
```

Programma 2 Conteggio di una lista in ingresso

Stadio 2 La tabella avrà la forma già vista nella risposta al TEST 1. Un semplice PRINT ...,... può essere sufficiente per visualizzare la tabella.

```
100 REM *VISUALIZZA LA TABELLA*
110 PRINT
120 PRINT
130 PRINT "INDICE","VOCE"
140 LET C=1
160 PRINT C,L(C)
170 LET C=C+1
180 IF C<=5 THEN GOTO 160
```

stampa la tabella

Programma 3 Stampa della lista in ordine di inserimento

Stadio 3 Ora facciamo in modo che C conti da 5 a 1 per stampare la lista in ordine inverso.

```
200 REM **STAMPA LA LISTA
         AL CONTRARIO**
210 PRINT
220 PRINT
230 LET C=5
250 PRINT L(C)
260 LET C=C-1
270 IF C>=1 THEN GOTO 250
```

stampa la lista in ordine inverso

Programma 4 Stampa in ordine inverso

Abbiamo visto i tre moduli di programma che forniscono la soluzione. Tutto ciò che dobbiamo fare adesso è di metterli insieme come segue. I pochi cambiamenti (che non intaccano quello che fa il programma) sono descritti nei commenti.

Cambiamenti

```
 10 REM **INSERIRE UNA LISTA**
 20 DIM L(5)
 30 LET C=1
 40 PRINT "INSERIRE IL PROSSIMO
NUMERO ";
 45 INPUT L(C)
 50 PRINT L(C)
 60 LET C=C+1
 70 IF C<=5 THEN GOTO 40
 80 REM
 90 REM
100 REM **VISUALIZZA LA
         TABELLA**
110 PRINT
120 PRINT
130 PRINT "INDICE","VOCE"
140 LET D=1
160 PRINT D,L(D)
170 LET D=D+1
180 IF D<=5 THEN GOTO 160
190 REM
195 REM
200 REM **STAMPA LA LISTA
         AL CONTRARIO**
210 PRINT
220 PRINT
230 LET E=5
250 PRINT L(E)
260 LET E=E-1
270 IF E>=1 THEN GOTO 250
280 STOP
```

Istruzioni REM per aiutare il lettore a vedere le divisioni nel programma.

Abbiamo usato D come indice per ricordarvi che il suo nome non ha significato - solo il suo valore.

Altri REM

E qui abbiamo usato E.

STOP aggiunto.

Programma 5 L'intero programma di inversione liste

```
RUN
INSERIRE IL PROSSIMO NUMERO -8
INSERIRE IL PROSSIMO NUMERO 15
INSERIRE IL PROSSIMO NUMERO 23
INSERIRE IL PROSSIMO NUMERO -4
INSERIRE IL PROSSIMO NUMERO 19

INDICE             VOCE
1                  -8
2                  15
3                  23
4                  -4
5                  19

19
-4
23
15
-8
```

K Programma 5.

Per risolvere il problema nell'Esempio 3 abbiamo fatto molta programmazione ma il programma lavora solo con una lista di cinque numeri. Un piccolo risultato per un grande sforzo se cambiamo le istruzioni da 20 a 70 che contano le cinque voci, possiamo fare in modo che il programma accetti qualsiasi numero di voci.

Ma abbiamo bisogno di rendere DIM L dipendente dal numero di voci che dobbiamo inserire.
Poiche il contatore C arriverà fino a N+1 (osservate la riga 70$), DIM L deve essere N+1.

```
  10 REM **INSERIRE UNA LISTA
           DI N VOCI **
  20 PRINT "QUANTI NUMERI NELLA
LISTA? ";
  21 INPUT N
  22 PRINT N
```

ora può essere inserito qualsiasi numero di voci nella lista L(C)

```
 24 DIM L(N+1)
 30 LET C=1
 40 PRINT "INSERIRE IL PROSSIMO
NUMERO ";
 45 INPUT L(C)
 50 PRINT L(C)
 60 LET C=C+1
 70 IF C<=N THEN GOTO 40
```

Cambiate inoltre, '5' con 'N' nelle righe 180 e 230.

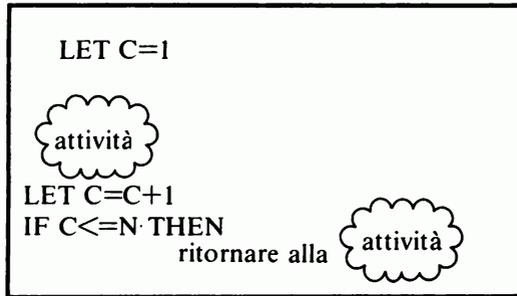
Programma 6

Essere in grado di inserire qualsiasi numero di dati dentro individuali locazioni di memoria con solo mezza dozzina di istruzioni rappresenta un significativo miglioramento nelle tecniche di programmazione. Naturalmente, non siamo mai soddisfatti perché dovremmo preoccuparci di contare le voci, specialmente se la lista è lunga, quando possiamo dire al computer di farlo per noi? Il prossimo esercizio vi chiede di fare questo.

Esercizio 1

Scrivete un programma BASIC per (a) inserire una lista di numeri di lunghezza sconosciuta, terminare la lista con il valore fittizio '-9999'. Chiamate questa lista P(C) e supponete che la lista sia di 30 voci o meno. Così DIM P(31) dichiarerà la lista. Verificate la vostra risposta. (b) Ora modificate il vostro programma per stampare quelle voci il cui indice è dispari.

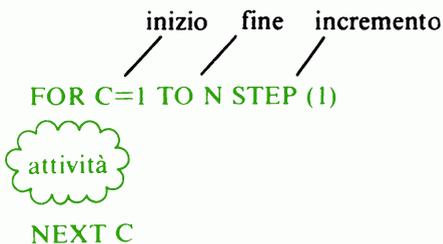
4.5 Il ciclo FOR ... NEXT ...

Come abbiamo detto ripetutamente un computer è ottimo per eseguire delle operazioni ripetitive. Per controllare queste operazioni dobbiamo usualmente contarle. Da quando abbiamo introdotto l'idea del conteggio nell'Unità 2, abbiamo usato diverse volte la seguente sequenza di istruzioni.

Programma 7

Queste operazioni ripetitive sono così importanti nella programmazione che per esse vengono usate delle specifiche istruzioni. In BASIC ciò viene realizzato dalle istruzioni FOR ... NEXT ...

La sequenza nel Programma 7 ha tre elementi:

LET C=1	che inizia il conteggio
LET C=C+1	che definisce i passi di incremento (1 in questo caso; 2 nella riga 270 della risposta all'Esercizio 1)
C<=N	che arresta il processo di conteggio

Le stesse prestazioni sono date dal ciclo FOR ... NEXT ... nel quale la sequenza descritta diventa:

Notate che NEXT C ritorna il controllo alla linea FOR C=1 TO N STEP (1) senza che dobbiate inserire il numero di linea di FOR C ... nell'istruzione NEXT C.

I vettori ed i cicli FOR ... NEXT sono fatti l'uno per l'altro. Insieme essi formano probabilmente la combinazione più potente del linguaggio BASIC. Vediamo in dettaglio come funzionano questi cicli, e quindi ripetiamo alcune delle nostre prime routines con le liste usando questa nuova prestazione.

Esempi di cicli FOR ... NEXT in azione

(a)

```
10 FOR I=4 TO 10 STEP (2)
20 PRINT I
30 NEXT I
40 STOP
4
6
8
10
```

(b)

```
10 FOR J=-3 TO 10 STEP (3)
20 PRINT J
30 NEXT J
40 STOP
-3
0
3
6
9
```

(c)

```
10 FOR K=11 TO 4 STEP (-2)
20 PRINT K
30 NEXT K
40 STOP
11
9
7
5
```

(d)

```
10 FOR L=4 TO -5 STEP (-2)
20 PRINT L
30 NEXT L
40 STOP
4
2
0
-2
-4
```

Programmi 9-12

TEST 2

Scrivete le liste di numeri prodotte dai seguenti cicli.

(a)

```
10 FOR E=1 TO 9 STEP (2)
20 PRINT E
30 NEXT E
40 STOP
```

(c)

```
10 FOR G=8 TO -4 STEP (-5)
20 PRINT G
30 NEXT G
40 STOP
```

(b)

```
10 FOR F=-30 TO -18 STEP (3)
20 PRINT F
30 NEXT F
40 STOP
```

(d)

```
10 FOR H=-2 TO -11 STEP (-4)
20 PRINT H
30 NEXT H
40 STOP
```

Programmi 13-16

FOR ... NEXT ... con STEP (1)

Gli esempi e i TEST hanno steps (passi) di 2, 3, -2, -4 e -5. Abbastanza spesso però vogliamo semplicemente usare uno step di 1. In questo caso, possiamo omettere STEP (1) dall'istruzione. Per cui

FOR C=1 TO N

NEXT C

Programma 17

è interpretato dal computer come uno step di 1.

Routines di inserimento/stampa con FOR ... NEXT

Le routines sono realizzate facilmente con FOR ... NEXT. Per esempio, possiamo riscrivere il Programma 5 usando tali cicli. Notate che alle righe 40 e 160 è richiesto uno step di 1 così STEP (1) è stato omesso.

Confronto con il Programma 5

```
 10 REM **INSERIRE UNA LISTA
           DI 5 NOMI **
 20 DIM L$(5,10)
 40 FOR C=1 TO 5
 50 PRINT "INSERIRE IL PROSSIMO
NOME ";
 55 INPUT L$(C)
 60 PRINT L$(C)
 70 NEXT C
 80 REM
 90 REM
100 REM *VISUALIZZA LA TABELLA*
110 PRINT
120 PRINT
130 PRINT "INDICE","VOCE"
140 PRINT
160 FOR D=1 TO 5
170 PRINT D,L$(D)
180 NEXT D
190 REM
195 REM
200 REM **STAMPA LA LISTA IN
           ORDINE INVERSO **
210 PRINT
220 PRINT
240 FOR E=5 TO 1 STEP (-1)
250 PRINT E,L$(E)
260 NEXT E
270 STOP
```

Sostituisce le righe da 20 a 70

Sostituisce le righe da 140 a 180

Sostituisce le righe da 230 a 270

Programma 18 Uso di FOR ... NEXT per invertire una lista

RUN

```
INSERIRE IL PROSSIMO NOME LAURA
INSERIRE IL PROSSIMO NOME FRANCO
INSERIRE IL PROSSIMO NOME ANNA
INSERIRE IL PROSSIMO NOME LISA
INSERIRE IL PROSSIMO NOME MARIO
```

```
INDICE              VOCE

1                   LAURA
2                   FRANCO
3                   ANNA
4                   LISA
5                   MARIO

5                   MARIO
4                   LISA
3                   ANNA
2                   FRANCO
1                   LAURA
```

K Programma 18.

Ciclo FOR ... NEXT in generale

Il ciclo FOR ... NEXT può essere espresso in termini generali come

```
FOR I = S TO F STEP (J)
  attività
NEXT I
```

provvedendo che ad S, F, J siano dati dei valori 'ragionevoli' prima che il ciclo venga eseguito.
Valori non ragionevoli potrebbero essere del tipo di

S=2, F=10, e STEP (-3)

poiché non potete ottenere dei numeri da 2 a 10 in steps di -3, nel normale corso degli eventi.

Esercizio 2

Se siete interessati alla matematica potreste voler fare il seguente programma che dimostra la potenza del ciclo FOR ... NEXT. Scrivete un programma per stampare in forma di tabella il quadrato ed il cubo di tutti i numeri interi dispari da 1 a 21 compreso.

Visualizzazione

Miriamo sempre ad una chiara presentazione dei dati in uscita sullo schermo o sulla stampante. L'istruzione FOR ... NEXT è usata largamente nelle routine di presentazione dei dati in uscita.

Uso per saltare delle righe. Come abbiamo visto nel programma per scrivere lettere nell'Unità 3 è utile 'stampare' delle linee vuote. La seguente routine fa questo per voi.

```
10 REM **FOR...NEXT**
20 REM **SALTO DI LINEE IN UNA
        ROUTINE DI STAMPA **
30 PRINT "SALVE"
40 FOR H=1 TO 10          istruzioni per stampare 10 linee vuote
50 PRINT
60 NEXT H
70 PRINT "SALVE DA 11 LINEE PI
U IN BASSO"
80 STOP
```

Programma 19

```
RUN
SALVE



SALVE DA 11 LINEE PIU IN BASSO
```

K Programma 19.

Disegno di una linea. Noi possiamo volere stampare delle linee attraverso lo schermo o pagina, per es. per separare dei blocchi di dati o solo per sottolineare. La seguente routine fa questo.

```
10 REM **FOR...NEXT**
20 REM **PER DISEGNARE LINEE**
40 FOR M=1 TO 32 ┐
50 PRINT "*";    ├──── istruzioni per stampare '*' 32 volte
60 NEXT M        ┘
70 PRINT
80 STOP
```

```
********************************
```

TEST 3

Perché la riga 50 nel Programma 20 ha; alla fine della riga? Cosa accade se la riga 50 è:

```
50 PRINT "*";
```

K Programma 20.

I cicli nei diagrammi di flusso

I cicli sono così importanti che hanno nei diagrammi di flusso un simbolo speciale a se stante:

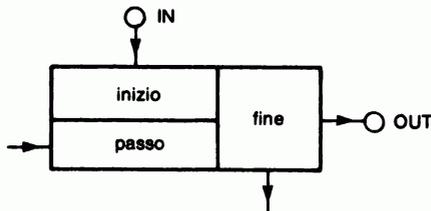



Figura 4a Simbolo nei diagrammi di flusso per il ciclo FOR ... NEXT ...

Le due parti terminali libere sono le connessioni con una attività:

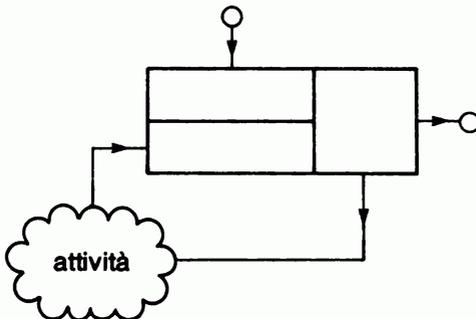



Figura 4b Simbolo dei diagrammi di flusso in relazione con una attività

4.6 Cicli nidificati

Il programma 20 disegnava una linea di 32 asterischi. Possiamo riscrivere il programma in modo da specificare nella riga 40 un numero di asterischi e variare così la lunghezza della linea. Così con il programma:

```
10 REM **LINEE DI DIVERSA
       LUNGHEZZA **
25 LET N=1   ——— numero che va alla riga 40
40 FOR M=1 TO N
50 PRINT "*";
60 NEXT M
70 PRINT
80 STOP
```

Programma 21

otteniamo:

```
RUN
*
```

Per variare la lunghezza della linea semplicemente inseriamo

25 LET N= lunghezza richiesta

e premiamo RUN:

```
25 LET N=2

RUN

**

25 LET N=3

RUN

***

25 LET N=4

RUN

****
```

```
25 LET N=8
```

RUN

```
********
```

```
25 LET N=16
```

RUN

```
****************
```

```
25 LET N=32
```

RUN

```
********************************
```

K Programma 21.

Cicli FOR ... NEXT ... nidificati

Avete visto nel Programma 21 in che modo potete controllare l'effetto del ciclo FOR ... NEXT ... delle righe da 40 a 60 cambiando il valore di N. Ora può venirvi in mente una domanda ovvia: 'Perché non controllare il valore di N usando un'altro ciclo FOR ... NEXT ...?' Il seguente programma fa appunto questo. Il ciclo-M delle righe da 50 a 70 è a sua volta controllato dal ciclo-N delle righe da 30 a 100. Il ciclo-M si dice essere 'nidificato' dentro il ciclo-N.

Programma 22 Cicli nidificati

RUN

```
*
**
***
****
*****
******
*******
********
*********
**********
***********
************
*************
**************
***************
****************
```

È importante che comprendiate come ciò avviene. Per esempio, quando il computer ha appena finito di stampare la 9° linea di asterischi, N sarà 9. Il programma lascia il ciclo interno (riga 50) ed il controllo va al ciclo esterno (riga 90). Viene stampata una linea vuota ed N viene incrementato al suo valore successivo di 10. Il controllo ritorna al ciclo interno alla riga 30. Il computer quindi gira intorno al ciclo interno (righe da 50 a 70) per dieci volte prima di uscire di nuovo alla riga 90.

K Programma 22.

Altri tipi di stampa con cicli

Se vi interessa produrre dei modelli di stampa con dei cicli, qui ce n'è un'altro, più due Esercizi.

```
 10 REM **CICLI FOR...NEXT
       NIDIFICATI **
 20 LET P=1
 40 FOR N=1 TO 5
 50 LET P=2*P
 60 PRINT
 70 FOR M=1 TO P
 80 PRINT "*";
 90 NEXT M
110 PRINT
120 NEXT N
140 STOP
```

Moltiplica per 2 il valore corrente di p ad ogni passaggio nel ciclo

```
RUN
**                                  = 2
****                                = 4
********                            = 8
****************                    =16
********************************    =32
```

K Programma 23.

Esercizio 3

Scrivete un programma usando dei cicli nidificati per stampare le tavole di moltiplicazione del 7, 8 e 9.

Esercizio 4

Scrivete un programma usando dei cicli nidificati per stampare rettangoli di asterischi di dimensioni scelte da voi.

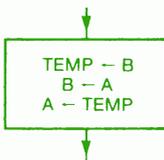
4.7 Interscambio

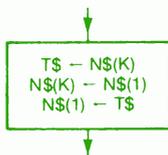
Abbiamo considerato il problema di trovare il più piccolo tra 2 numeri nell'Unità 2, ed il più piccolo di 3 nell'Unita 3. In questa Unità abbiamo fatto degli esercizi sullo scambio di elementi di una lista, come preparazione per scrivere un programma di interscambio.

Abbiamo confrontato i numeri di una lista con il numero che si trovava alla posizione 1, scambiandoli se il primo numero nella lista era più piccolo di quello nella posizione 1. Quando avete fatto ciò nel TEST 1, lo avete fatto manualmente, non abbiamo ancora considerato il problema di scrivere un programma per eseguire lo scambio. Non possiamo dire solo:

Copia A in B e quindi B in A

poiché la prima transazione 'copia A in B' sovrascrive e distrugge ciò che si trova in B, dandoci delle copie di A in A ed in B. Dobbiamo mettere da parte il contenuto di B in una memoria temporanea **prima** di sovrascrivere B con A. Vediamone schematicamente il processo:

TEMP ← B
B ← A
A ← TEMP

Dove ←: 'significa sposta il numero nella locazione a destra nella locazione di sinistra'.

Supponiamo, che cerchiate di riordinare una lista di nomi di nome N$, interscambiando il primo nome, N$(1), con qualcun'altro nome, N$(K) alla posizione K. Ciò può essere fatto usando una locazione di memoria temporanea T$:

Ricordate che è il contenuto di N$(1) ed N$(K) che viene scambiato.
Supponiamo che N$(1)=FRANCO ed N$(K)=GIANNI, questo è ciò che avviene:

	Locazioni di memoria		
	N$(1)	N$(K)	T$
inizio	FRANCO	GIANNI	
prossimo stadio	FRANCO	GIANNI	GIANNI
prossimo stadio	FRANCO	FRANCO	GIANNI
fine	GIANNI	FRANCO	GIANNI

(Il fatto che T$ conservi ancora 'GIANNI' non ha importanza; abbiamo raggiunto l'obiettivo che era di scambiare le locazioni di FRANCO e GIANNI.)

Diagramma di flusso per riordino nomi

Nel riordino di numeri (TEST 1) abbiamo messo il più basso al vertice della lista. Allo stesso modo confrontiamo a turno ciascun nome con quello che si trova in quel momento al vertice della lista e li scambiamo solo se il nome confrontato viene prima di quello al vertice della lista. Un diagramma di flusso per questo è:

Il contatore inizia da 2

Confronto

Scambio se necessario

Aggiungere 1 al contatore

Finito?

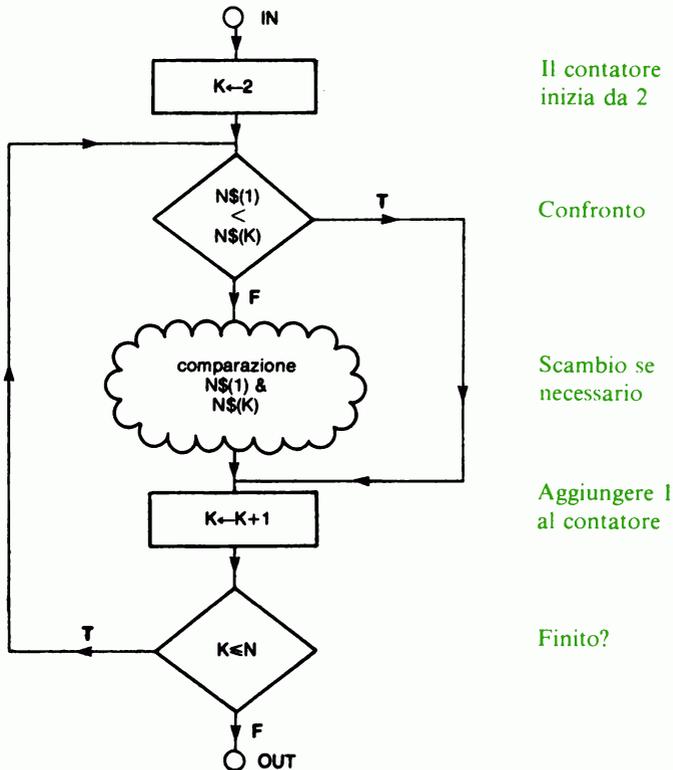



Figura 5a Diagramma di flusso per interscambio

Se vogliamo usare il simbolo speciale per FOR ... NEXT ..., dovrebbe apparire come:

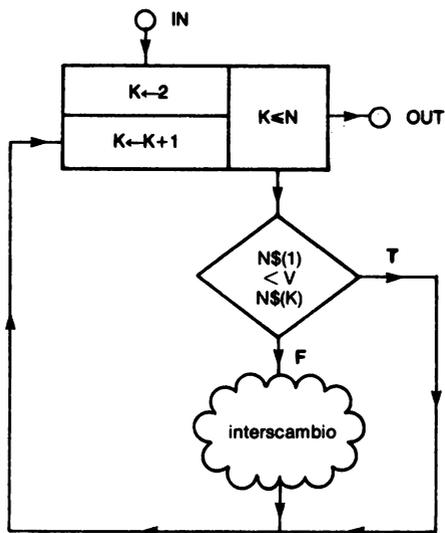



Figura 5b Diagramma di flusso per interscambio con il simbolo di FOR ... NEXT ...

Questa nuova routine può ora essere usata per costruire un programma.

Esempio 4

Scriviamo un programma per inserire una lista di nomi di lunghezza sconosciuta in un vettore, stampare questa lista con l'indice in ordine di inserimento. Per mezzo della routine di interscambio portiamo il nome di valore alfabetico più basso nella posizione 1 della lista, e visualizziamo la lista risultante.

Soluzione

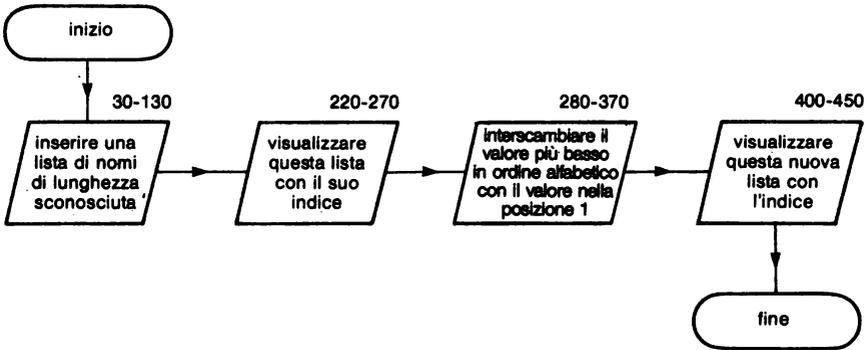



Figura 6 Diagramma di flusso per l'Esempio 4

Programma di interscambio

```
  10 REM **PRIMO IN ORDINE
          ALFABETICO **
  20 CLEAR
  25 DIM N$(20,6)
  30 PRINT "INSERIRE UNA LISTA D
I NOMI UNO  PER UNO"
  40 PRINT "TERMINARE LA LIST CO
N ZZZZ"
  50 PRINT
  60 LET I=1
  70 PRINT "PROSSIMO NOME? ";
  71 INPUT N$(I)
  72 PRINT N$(I)
  80 IF N$(I)="ZZZZ  " THEN GOTO
 200
  90 LET I=I+1
 100 GOTO 70
 180 REM ***********************
 190 REM **ZZZZ NON NECESSARIO
          NELLA LISTA, COSI**
 200 LET N=I-1
 210 REM ***********************
 300 PRINT
 310 PRINT "INDICE","VOCE"
 320 FOR J=1 TO N
 330 PRINT J,N$(J)
 340 NEXT J
 400 REM ***********************
```

inserimento lista

calcolo del numero di nomi inseriti

stampa la lista

```
410 REM ROUTINE DI INTERSCAMBIO
420 FOR K=2 TO N
430 IF N$(1)<N$(K) THEN GOTO 47
0
440 LET T$=N$(K)
450 LET N$(K)=N$(1)
460 LET N$(1)=T$
470 NEXT K
500 REM ************************
510 PRINT
520 PRINT "LISTA DOPO LO SCAMBI
O"
530 PRINT
540 PRINT "INDICE","VOCE"
550 FOR L=1 TO N
560 PRINT L,N$(L)
570 NEXT L
580 STOP
```


Programma 24 Ricerca della prima voce in ordine alfabetico

Notate che nella riga 130 vi sono due spazi dopo 'ZZZZ'; è importante!

Funzionamento del programma

RUN

```
INSERIRE UNA LISTA DI NOMI UNO
PER UNO
TERMINARE LA LIST CON ZZZZ

PROSSIMO NOME? GIANNI
PROSSIMO NOME? PAOLA
PROSSIMO NOME? DAVIDE
PROSSIMO NOME? ENRICO
PROSSIMO NOME? ZZZZ

INDICE          VOCE
1               GIANNI
2               PAOLA
3               DAVIDE
4               ENRICO

LISTA DOPO LO SCAMBIO

INDICE          VOCE
1               DAVIDE
2               PAOLA
3               GIANNI
4               ENRICO
```

Se viene inserita una routine di stampa come quella qui in basso nella procedura di scambio (righe da 342 a 360), possiamo osservare l'effetto di ciascun passaggio attraverso il ciclo.

```
400 REM **********************
410 REM ROUTINE DI INTERSCAMBIO
420 FOR K=2 TO N
430 IF N$(1)<N$(K) THEN GOTO 47
0
440 LET T$=N$(K)
450 LET N$(K)=N$(1)
460 LET N$(1)=T$
470 PRINT
472 FOR L=1 TO N
474 PRINT N$(L);"  ";
476 NEXT L
478 PRINT
480 NEXT K
500 REM **********************
```

routine di stampa per osservare il passaggio attraverso il ciclo

K Programma 24 con le righe da 400 a 500 come sopra.

Obiettivi del capitolo 4

Ora che avete completato questa Unità, verificate di essere in grado di scrivere dei semplici programmi usando:

Liste di locazioni di memoria
- per inserire liste ☐
- per stampare liste ☐

Contatori per contare il numero di voci in una lista e

Cicli FOR ... NEXT ...
- per stampare una lista ☐
- per inserire una lista ☐
- per stampare gruppi di * ☐

Cicli nidificati
- per stampare gruppi di * ☐

Routine di interscambio ☐

Risposte ai TEST ed agli Esercizi

TEST 1

I sei stadi della procedura sono mostrati qui di seguito nell'esempio di funzionamento:

```
RUN

INSERIRE UNA LISTA DI NOMI UNO
PER UNO
TERMINARE LA LISTA CON ZZZZ

PROSSIMO NOME? 6
PROSSIMO NOME? 8
PROSSIMO NOME? 4
PROSSIMO NOME? 7
PROSSIMO NOME? 3
PROSSIMO NOME? 9
PROSSIMO NOME? 1
PROSSIMO NOME? ZZZZ

INDICE          VOCE
1               6
2               8
3               4
4               7
5               3
6               9
7               1

6    8    4    7    3    9    1

4    8    6    7    3    9    1

4    8    6    7    3    9    1

3    8    6    7    4    9    1

3    8    6    7    4    9    1

1    8    6    7    4    9    3
```

```
LISTA DOPO LO SCAMBIO

INDICE              VOCE
1                   1
2                   8
3                   6
4                   7
5                   4
6                   9
7                   3
```

Esercizio 1

Notate che abbiamo usato molte istruzioni REM per dirvi come lavora il programma.

```
   5 DIM P(31)
  10 REM *UNA LISTA DI NUMERI DI
          LUNGHEZZA SCONOSCIUTA
  20 PRINT "INSERIRE GLI ELEMENT
I DELLA "
  22 PRINT "LISTA UNO PER UNO CO
ME RICHIESTO"
  24 PRINT "TERMINARE LA LISTA C
ON -9999"
  26 PRINT
  30 LET C=1                          ┐
  40 PRINT "INSERIRE IL PROSSIMO      │
 NUMERO ";                            ├─Sequenza di inserimento
  41 INPUT P(C)                       │
  42 PRINT P(C)                       │
  50 IF P(C)=-9999 THEN GOTO 100      ┘
  60 LET C=C+1
  70 GOTO 40
  80 REM ***********************
  90 REM ** C HA CONTATO -9999
           COME UN ELEMENTO **
 100 LET N=C-1 ]──────────────────── Prende il totale corretto
 110 REM ***********************      dal contatore
 120 REM
 130 REM
 200 REM **STAMPA GLI ELEMENTI
          CUI INDICE E DISPARI*
 210 REM **3=1+2..5=3+2...ETC.**
 220 LET C=1
 230 PRINT
 240 PRINT
 250 PRINT "INDICE DISPARI","ELE
MENTO"
```

```
260 PRINT C,P(C)
270 LET C=C+2
280 IF C<N THEN GOTO 260
290 STOP
```

Sequenza di stampa

Programma 25

RUN

```
INSERIRE GLI ELEMENTI DELLA
LISTA UNO PER UNO COME RICHIESTO
TERMINARE LA LISTA CON -9999

INSERIRE IL PROSSIMO NUMERO 42
INSERIRE IL PROSSIMO NUMERO -12
INSERIRE IL PROSSIMO NUMERO 37
INSERIRE IL PROSSIMO NUMERO 92
INSERIRE IL PROSSIMO NUMERO 11
INSERIRE IL PROSSIMO NUMERO -3
INSERIRE IL PROSSIMO NUMERO -999
9

INDICE DISPARI   ELEMENTO
1                42
3                37
5                11
```

TEST 2

(a) 1,3,5,7,9.

(b) -30,-27,-24,-21,-18.

(c) 8,3,-2.

(d) -2,-6,-10.

Esercizio 2

```
 10 REM **QUADRATI E CUBI **
 40 PRINT "NUMERO----QUADRATO--
--CUBO"
 60 FOR I=1 TO 21 STEP (2)
 70 LET S=I*I
 80 LET C=I*I*I
 90 PRINT I;
100 PRINT "           ";S;
110 PRINT "             ";C
120 NEXT I
130 STOP
```

RUN

```
NUMERO----QUADRATO----CUBO
1         1           1
3         9           27
5         25           125
7         49          343
9         81          729
11          121          1331
13          169          2197
15          225          3375
17          289          4913
19          361          6859
21          441          9261
```

La stampa è un po' disordinata, ma faremo di meglio nella prossima Unità usando TAB.

TEST 3

Sopprime il ritorno a capo durante la stampa così che il cursore si ferma dopo aver stampato *. Per cui il prossimo * sarà stampato sulla stessa linea. Senza ; gli asterischi sarebbero stampati in una colonna profonda 32 linee.

Esercizio 3

```
 10 REM **TABELLE DI
         MOLTIPLICAZIONE **
 40 FOR T=7 TO 9
 60 FOR K=1 TO 12
 70 LET P=K*T
 80 PRINT K;" VOLTE ";T;" = ";P
 90 NEXT K
110 PRINT
120 NEXT T
140 STOP
```

Programma 27

RUN

```
1 VOLTE 7 = 7
2 VOLTE 7 = 14
```

```
3 VOLTE 7 = 21
4 VOLTE 7 = 28
5 VOLTE 7 = 35
6 VOLTE 7 = 42
7 VOLTE 7 = 49
8 VOLTE 7 = 56
9 VOLTE 7 = 63
10 VOLTE 7 = 70
11 VOLTE 7 = 77
12 VOLTE 7 = 84

1 VOLTE 8 = 8
2 VOLTE 8 = 16
3 VOLTE 8 = 24
4 VOLTE 8 = 32
5 VOLTE 8 = 40
6 VOLTE 8 = 48
7 VOLTE 8 = 56
8 VOLTE 8 = 64
9 VOLTE 8 = 72
10 VOLTE 8 = 80
11 VOLTE 8 = 88
12 VOLTE 8 = 96

1 VOLTE 9 = 9
2 VOLTE 9 = 18
3 VOLTE 9 = 27
4 VOLTE 9 = 36
5 VOLTE 9 = 45
6 VOLTE 9 = 54
7 VOLTE 9 = 63
8 VOLTE 9 = 72
9 VOLTE 9 = 81
10 VOLTE 9 = 90
11 VOLTE 9 = 99
12 VOLTE 9 = 108
```

abbiamo ancora qualcosa
da imparare sulla 'tabulazione'

Esercizio 4

```
  10 REM **RETTANGOLO**
  20 PRINT "LUNGHEZZA DEL RETTAN
GOLO? ";
  30 INPUT L
  40 PRINT L
  50 PRINT "LARGHEZZA DEL RETTAN
GOLO? ";
  60 INPUT W
  70 PRINT W
  80 FOR I=1 TO L
 100 FOR J=1 TO W
 110 PRINT "*";
 120 NEXT J
 140 PRINT
 150 NEXT I
 170 STOP
```

Programma 28

```
LUNGHEZZA DEL RETTANGOLO? 10
LARGHEZZA DEL RETTANGOLO? 6
******
******
******
******
******
******
******
******
******
******
```

CAPITOLO 5

TUTTO SU STRINGHE E PRINT

5.1 Introduzione

Le unità iniziali toccavano gli argomenti introduttivi; le nuove idee venivano trattate rapidamente ed in modo generale. Questa unità tratta principalmente delle stringhe, ma incontrerete anche l'istruzione TAB che è un'aggiunta importante alle vostre conoscenze sulla stampa. Il titolo di questa unità è una leggera esagerazione, ma alla fine di questo capitolo avrete fatto la conoscenza con la maggior parte delle principali funzioni di stringa e di stampa del linguaggio BASIC.

5.2 Lunghezza di una stringa di caratteri

Nell'Unità 3 ci domandammo 'Quanto è lunga una stringa?'. In quel momento poteva sembrare una domanda inutile, ma il numero di caratteri contenuto in una particolare locazione di stringa è spesso una informazione vitale. Questo specialmente se cerchiamo di usare le locazioni di memoria di un particolare computer nel modo più efficente possibile.

Nel BASIC l'operazione LEN A$) dà la lunghezza di A$ come numero di caratteri. Per cui

Se A$="LISA" allora LEN (A$)= 4
Se B$="I" allora LEN (B$) =

TEST 1

Quali sono i valori delle seguenti espressioni?:

(a) LEN (C$) dove C$="T"
(b) LEN (D$) dove D$="A"
(c) LEN (E$) dove E$="72"
(d) LEN (F$) dove F$="CAT 256"

Esempio 1

Scriviamo un programma BASIC per inserire una lista di otto parole, terminante con 'ZZZZ', e stampare la lunghezza di ciascuna parola.

Soluzione

```
10 REM **LUNGHEZZA DI UNA
       PAROLA **
20 REM ************************
30 REM **INSERISCE LE PAROLE
       UNA AD UNA E STAMPA
       LA LORO LUNGHEZZA   **
60 LET I=0
70 LET I=I+1
80 PRINT "PROSSIMA PAROLA"
```

```
100 INPUT W$
110 IF W$="-9999" THEN GOTO 200
120 LET L=LEN W$
130 PRINT
140 PRINT W$;" HA ";L;" LETTERE
145 IF I=10 THEN GOTO 200
150 GOTO 70
200 STOP
```

Programma 1 Misurare la lunghezza delle parole

```
PROSSIMA PAROLA

SVILUPPATE HA 10 LETTERE
PROSSIMA PAROLA

UN HA 2 LETTERE
PROSSIMA PAROLA

ALGORITMO HA 9 LETTERE
PROSSIMA PAROLA

E HA 1 LETTERE
PROSSIMA PAROLA

SCRIVETE HA 8 LETTERE
PROSSIMA PAROLA

UN HA 2 LETTERE
PROSSIMA PAROLA

PROGRAMMA HA 9 LETTERE
PROSSIMA PAROLA

BASIC HA 5 LETTERE
```

(Risultato dopo aver inserito le parole: SVILUPPATE, UN, ALGORITMO, E, SCRIVETE, UN, PROGRAMMA, BASIC, ZZZZ.)

K Programma 1.

5.3 Tabelle di frequenza

Misurare la frequenza con cui avvengono alcune cose è comunemente necessario nel trattamento di informazioni numeriche. Per esempio, la conoscenza della frequenza con cui certe lettere capitano nel normale uso del linguaggio è un fattore importante nelle attività di decodifica di crittogafie. Per misurare le frequenze è possibile usare dei semplici metodi di analisi statistica. Il primo esempio (fatto con carta e matita) ci introduce a ciò.

Segni di riscontro

Esempio 2

Troviamo la frequenza con cui ciascuna vocale capita nelle seguenti parole:

BIT, COMPUTER, MEMORIA, AEREO, VOLO, BASIC, USA, VINO, VENEZIA, USCITA, MELE, PUNTI, U, SALE, SENEGAL, ORO, VERI, LUCE, TERRE, MATITE, MOLO, MITE

Soluzione

Ci sono due modi di affrontare il problema.

(a) Passare attraverso le parole segnando e contando tutte le A, quindi ripassare per contare le E, ecc. Questo richiede 5 passaggi attraverso i dati per informazioni abbastanza sparse;

(b) Disegnare una tabella come quella in basso e mettervi ciascuna vocale in sequenza:

BIT: mettere un segno di riscontro nella riga delle I;

COMPUTER: mettere un segno nella riga delle O uno nella riga delle U ed uno nella riga delle E;

MEMORIA: mettere un segno ciascuna nelle righe di E, O, I, A.

Vocali	*Conteggio*				*Conteggio totale o frequenza*
A	𝍸	1111			9
E	𝍸	𝍸	𝍸	1	16
I	𝍸	𝍸			10
O	𝍸	𝍸			10
U	𝍸	1			6

BI̸T, CØMPU̸TE̸R, ME̸MØRI̸A̸, A̸E̸RE̸Ø, VØLØ, BA̸SI̸C, U̸SA̸, VI̸NØ, VE̸NE̸-ZI̸A̸, U̸SCI̸TA̸, ME̸LE̸, PU̸NTI̸, U̸, SA̸LE̸, SE̸NE̸GA̸L, ØRØ, VE̸RI̸, LU̸CE̸, TE̸RRE̸, MA̸TI̸TE̸, MØLØ, MI̸TE̸.

Figura 1 Conteggio di riscontro completato

TEST 2

Usate il metodo descritto per disegnare una tabella di frequenza della lunghezza delle parole indicate nell'Esempio 2.

Fare contare il computer

Trovato un metodo di contare la frequenza con carta e matita, ora noi cerchiamo un metodo per indurre il computer a fare il conteggio di una lista. La potenza di una lista deriva da un uso appropriato dell'indice.

Supponiamo, di volere contare il numero di volte che le cifre 0, 1, 2, ... 9 capitano in una sequenza. Possiamo usare dieci contatori:

C(0), C(1), C(2) ... C(9)

ciascuno dei quali all'inizio sarà a zero. Per contare le cifre in 473808 prendiamo la prima cifra nella sequenza: 4. 1 viene aggiunto a C(4) e così via:

Cifre inserite	Contatori dopo l'inserimento									
	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
inizio	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	1	0	0	1	0	0
3	0	0	0	1	1	0	0	1	0	0
8	0	0	0	1	1	0	0	1	1	0
0	1	0	0	1	1	0	0	1	1	0
8	1	0	0	1	1	0	0	1	2	0

Così l'idea è che quando alla tastiera viene inserito I, incrementi C(I) di 1. Questo può essere ottenuto con solo due istruzioni BASIC.

```
120 INPUT I
140 LET C(I)=C(I)+1
```

Un contatore di lista

Tuttavia, lo ZX81 non ci consente di usare C(0) come locazione - le locazioni devono partire da 1, per es. C(1), C(2), C(3), ... ecc. Noi possiamo aggirare il problema usando

C(1) per contare gli 0
C(2) per contare gli 1
C(3) per contare i 2 ecc.

Per cui C(N) conta N-1 così quando inseriamo N noi incrementiamo il contatore C(N+1).

TEST 3

La sequenza

```
10 INPUT N
20 LET C(N+1)=C(N+1)+1
```

è usata per contare il numero di 0, 1, 2, ecc. nei seguenti dati inseriti: 3, 1, 0, 5, 9, 9, 6, 6, 6, 0, 4, 4, 2, 4, 1, 2, 1, 3, 0, 2, 1, 3. Quali sono i valori dei seguenti contatori?

(a) C(4) dopo che sono stati inseriti 3 numeri.
(b) C(10) dopo che sono stati inseriti 12 numeri.
(c) C(2) dopo che sono stati inseriti tutti i numeri.
(d) C(1) dopo che sono stati inseriti tutti i numeri.

Ora useremo questo metodo di conteggio di una lista in un esempio.

Esempio 3

Scriviamo un programma per inserire una sequenza di cifre singole e fornire la frequenza con cui capita ciascuna cifra.

Soluzione

Una cifra è uno dei numeri della serie 0, 1, 2, 3, ... 9. Noi inseriremo queste cifre una ad una, e la sequenza sarà terminata da -9999.

La lista di conteggio avrà dieci contatori:

C(1), C(2), C(3) ... C(10)

Il programma per risolvere il problema consiste di due parti. (i) La routine di inserimento ed incremento che incorpora le due istruzioni 120 e 140 discusse sopra. (ii) La routine di stampa controllata da un ciclo FOR ... NEXT ..., con l'indice J che va da 0 a 9.

```
 10 REM **CONTA IL NUMERO DI
          VOLTE IN CUI E INSERI
          TA CISCUNA CIFRA E LO
          MEMORIZZA IN UNA
          LISTA DI CONTEGGIO
          C(I) **
 30 REM ***********************
 40 DIM C(10)
100 PRINT "INSERIRE UNA LISTA D
```

```
I CIFRE"
 110 PRINT "FINIRE LA LISTA CON
-9999"
 115 PRINT
 120 PRINT "PROSSIMA CIFRA"
 121 INPUT I
 122 PRINT I
 130 IF I=-9999 THEN GOTO 200
 140 LET C(I+1)=C(I+1)+1
 150 GOTO 120
 190 REM ***********************
 200 CLS
 205 PRINT
 210 PRINT "CIFRA","CONTO"
 220 PRINT
 230 FOR J=1 TO 10
 240 PRINT J-1,C(J)
 250 NEXT J
 260 STOP
```

routine di inserimento e conteggio

stampa la tabella

Programma 2 Contare con una lsta di calcolo c(i)

Stampa tipo

(Dopo aver inserito 3, 7, 6, 4, 9, 1, 4, 9, 2, 7, 8, 0, 1, 5, 2, 7, -9999.)

```
CIFRA               CONTO

0                   1
1                   2
2                   1
3                   1
4                   2
5                   1
6                   1
7                   3
8                   1
9                   2
```

K Programma 2.

Tabella di frequenza per lunghezze di stringhe

In questa unità finora abbiamo scritto due programmi; il primo per trovare la lunghezza di stringhe, ed il secondo per costruire una tabella di frequenza.

Nel prossimo esercizio vorremmo che combiniate queste due idee per costruire una tabella di frequenza della lunghezza di parole. Se volete potete usare le parole già usate nell'Esempio 2. Date per scontato che le parole non saranno più lunghe di 15 caratteri, così la lista-lunghezze sarà formata dagli elementi:

L(1), L(2), L(3) ... L(15).

Esercizio 1

Scrivete un programma per inserire una serie di parole e visualizzare una tabella di frequenza delle loro lunghezze.

5.4 Diagrammi di frequenza

Diagramma di frequenza del numero di vocali

L'immagine schematica in Figura 1 produce su di noi un impatto più immediato e ci da più informazioni sulla distribuzione di frequenza delle vocali di una semplice colonna di numeri. Così perché non far disegnare al computer un immagine per noi? Avete visto nell'Unità 4 come stampare delle righe di asterischi usando l'istruzione PRINT ed i cicli FOR ... NEXT.

TEST 4

Cosa apparirà sullo schermo come risultato del seguente programma inserendo i numeri 2, 5, 7, 8, 3, 1?

```
10 INPUT A
20 FOR I=1 TO A
30 PRINT "*";
40 NEXT I
45 PRINT
50 GOTO 10
```

Programma 3

Possiamo fare la stessa cosa usando le frequenze di Figura 1 per stabilire il numero di asterischi da stampare sullo schermo. Questo produrrà un'immagine della distribuzione.

Esempio 4

Scriviamo un programma per stampare un diagramma di frequenza per la distribuzione di vocali data nell'Esempio 2.

Soluzione

Notate che questo programma disegna il diagramma a partire dalle frequenze che abbiamo già calcolato. Noi inseriamo tali frequenze con le linee da 40 a 80.

Leggiamo le frequenze (righe da 42 a 80) con un contatore F(K) dove F(1) è il numero di a, F(2) il numero di e, ecc.

Quindi stampiamo gli asterischi sullo schermo secondo i valori di F(K) (righe da 220 a 250).

```
 10 REM **DISTRIBUZIONE DI
        FREQUENZA **
 20 REM **PREPARAZIONE DEL
        DISEGNO**
 30 REM **LA LISTA DELLE FRE-
        QUENZE E F(K) **
 35 DIM F(6)
 40 LET K=1
 42 PRINT "VOCALE",K
 44 PRINT "INSERIRE LA FREQUENZ
A ";
 45 INPUT F(K)
 46 PRINT F(K)
 60 IF F(K)=-9999 THEN GOTO 100
 70 LET K=K+1
 80 GOTO 42
 90 REM ************************
100 REM **NON AGGIUNGERE -9999
        ALLA LISTA **
110 LET N=K-1
120 REM ************************
200 REM **ROUTINE DI STAMPA**
205 CLS
210 PRINT
220 FOR X=1 TO N
230 FOR Y=1 TO F(X)
240 PRINT "*";
250 NEXT Y
260 PRINT
270 PRINT
280 NEXT X
300 REM ************************
```

legge le frequenze e le memorizza in F(1), F(2) ... (righe 60–80)

stampa * sullo schermo (righe 210–280)

```
RUN
*********
****************
**********
**********
******
```

K Programma 4.

Diagramma di frequenza della lunghezza delle parole

Se vogliamo disegnare un diagramma della frequenza con cui le varie lunghezze di parola ricorrono nell'Esempio 2, abbiamo bisogno di modificare il Programma 4. Sono necessarie due modifiche:

Per prima cosa la lista delle frequenze deve contenere più elementi. Ci sono 15 frequenze (da 1 a 15) più -9999, quindi 16 elementi per cui aggiungiamo 35 DIM F(16) al Programma 4.

Per seconda, la routine di stampa alla riga 220 presenterà dei problemi quando la frequenza sarà zero. Noi non possiamo far funzionare il ciclo FOR ... NEXT da 1 a 0 perciò occorre evitare che il programma vada nel ciclo FOR ... NEXT quando la frequenza è zero. Per far ciò aggiungiamo 225 IF F(X)=0 THEN GO TO 260.

Dobbiamo anche modificare le istruzioni di ingresso in:

```
42 PRINT "LUNGH.",K
44 PRINT "INSERIRE FREQUENZA ";
```

Così il programma sarà

```
Programma 4 ——inserire  35 DIM F(16)
            ——inserire  42 PRINT "LUNGH.",K
            ——inserire  44 PRINT "INSERIRE FREQUENZA ";
            ——inserire 225 IF F(X)=0 THEN GOTO 260
```

Il funzionamento del Programma 4 modificato produce:

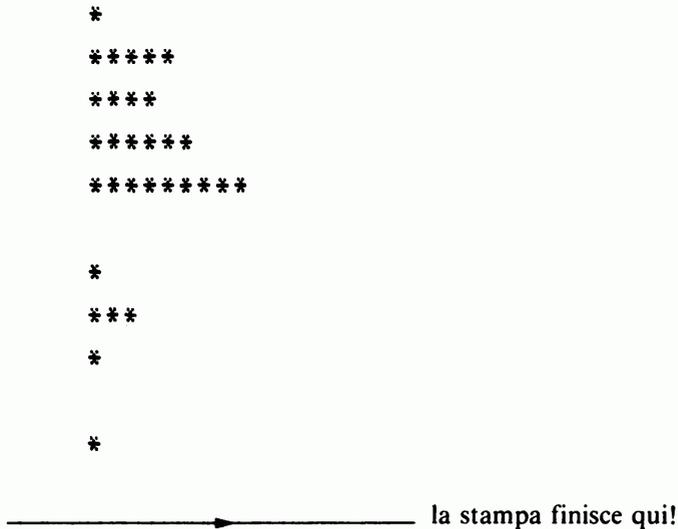



Figura 2 Diagramma di frequenza del Programma 4 modificato.

K Programma 4 (modificato).

5.5 Tabulazione

Abbiamo ottenuto gli ingredienti essenziali di un immagine, ma essa è ancora lontana dall'essere un diagramma significativo. Potrebbe aiutarci avere la possibilità di spostarci e stampare sullo schermo in qualsiasi posizione predeterminata. Nell'uso di una macchina da scrivere questo viene chiamato tabulazione (da tabulare o 'a forma di tabella'). Nel BASIC ciò viene realizzato dalla funzione TAB.

Adoperiamo lo stesso sistema usato nell'Unità 3, vale a dire scriviamo un frammento di programma che descrive se stesso.

Per prima cosa, osservate cosa appare se numerate le posizioni di stampa sullo schermo:

Programma 5

Potete vedere che TAB (5) ha stampato E alla sesta posizione. Perché? Perché la macchina conta le posizioni di stampa dalla posizione 0. Ciò è dimostrato dal Programma 6 nel quale la scala sullo schermo parte da 0:

```
  50 PRINT "0123456789012345678901234567
8901"
  60 PRINT "A";TAB (5);"E";TAB (7);
"I";TAB (19);"O";TAB (31);"U"
```

RUN

```
01234567890123456789012345678901
A    E I           O           U
```

Programma 6

Ora TAB (5) va alla posizione etichettata 5 ma essa è sempre la sesta posizione dello schermo.

Nota: Sullo ZX81, TAB A è sufficente per TABulare alla posizione A, ma molti computers richiedono anche le parentesi, vale a dire TAB (A). Perciò abbiamo messo le parentesi a tutte le istruzioni TAB.

TEST 5

Scrivete un programma per stampare COL 1, COL 2, COL 3 sullo schermo con COL 1 che inizi alla posizione 0, COL 2 alla posizione 10 e COL 3 alla posizione 20.

TAB variabile e suoi effetti

Possiamo pilotare la riga 60 della stampa vocali con un ciclo FOR ... NEXT per produrre una vera tabella.

```
  50 FOR I=1 TO 7
  60 PRINT "A";TAB (5);"E";TAB (
7);"I";TAB (19);"O";TAB (31);"U"
  70 NEXT I
```

Programma 7

RUN

```
A    E I           O           U
A    E I           O           U
A    E I           O           U
```

```
A    E I           O          U
A    E I           O          U
A    E I           O          U
A    E I           O          U
```

C'è un'altro esempio che mostra come possiamo pilotare TAB con una variabile. Se usiamo TAB (V) dove V è una variabile, possiamo muovere il cursore di stampa su differenti posizioni dello schermo. Il programma

```
30 FOR A=1 TO 10
40 PRINT TAB (A);"SALVE"
50 NEXT A
60 STOP
```

Il valore di A in TAB (A) è determinato dalla variabile di ciclo a.

Programma 8

produce

RUN

```
SALVE
 SALVE
  SALVE
   SALVE
    SALVE
     SALVE
      SALVE
       SALVE
        SALVE
         SALVE
```

Noi possiamo andare ancora un passo avanti e combinare questi due effetti in un solo programma:

```
  50 FOR I=0 TO 6
  60 PRINT TAB (0+I);"A";TAB (5+
I);"E";TAB (7+I);"I";TAB (19+I);
"O";TAB (24+I);"U"
  70 NEXT I
  80 STOP
```

RUN

TEST 6

Scrivete un segmento di programma per inserire tre numeri a vostra scelta che permettano di posizionare la stringa "TITOLO" in tre diverse posizioni della stessa riga di stampa.

TAB ed il diagramma di frequenza

Ora abbiamo la possibilità di realizzare il diagramma di frequenza di Figura 2 in maniera più elegante.

La routine di stampa del Programma 4 modificato (righe da 200 a 300) era:

```
200 REM **ROUTINE DI STAMPA**
205 CLS
210 PRINT
220 FOR X=1 TO N
225 IF F(X)=0 THEN GOTO 260
230 FOR Y=1 TO F(X)
240 PRINT "*";
250 NEXT Y
260 PRINT
270 PRINT
280 NEXT X
300 REM ************************
```

Programma 4 (modificato)

Aggiungiamo:

la riga 212 per stampare i titoli delle colonne

la riga 214 per stampare una linea attraverso lo schermo

la riga 216 per iniziare la divisione delle colonne (il resto delle divisioni viene stampato dai cicli seguenti)

la riga 222 (nel ciclo) per stampare X ed F(X) attraverso la pagina con in più le divisioni delle colonne. Questa linea finisce con ';' il quale fa in modo che la successiva istruzione PRINT (linea 240) appaia sulla stessa riga.

Probabilmente dovrete esaminare il programma con cura per scoprire tutti i dettagli in esso presenti:

```
200 REM **ROUTINE DI STAMPA**
205 CLS
210 PRINT
212 PRINT "LUNGH.";TAB (8);"FRE
Q.";TAB (18);"GRAFICO"
214 PRINT "--------------------
-----"
216 PRINT TAB (7);":";TAB (12);
":"
220 FOR X=1 TO N
222 PRINT TAB (2);X;TAB (7);":"
;TAB (9);F(X);TAB (12);":";TAB (
14);
225 IF F(X)=0 THEN GOTO 260
230 FOR Y=1 TO F(X)
240 PRINT "*";
250 NEXT Y
260 PRINT
270 PRINT
280 NEXT X
300 REM ***********************
```

Programma 10

K Programma 10

Esercizio 2

Modificate il programma 4 per ottenere una stampa simile a quella del Programma 10 e che includa i seguenti punti:
(a) un appropriato cambio di titoli;
(b) la stampa delle lettere A, E, I, O, U nella colonna a sinistra;
(c) una scala appropriata alla base del diagramma.

5.6 Suddividere le stringhe

Osserviamo ora una stringa che, sebbene sia una entità a se stessa, contiene più di un elemento di informazione. Per esempio, 23 Giugno 1971 è una singola data ma ci sono occasioni nelle quali noi cerchiamo solo una parte di essa, per es. il mese.

Archiviare le date

Quante volte vi siete trovati di fronte, per es. in un modulo da compilare, ad un riquadro come questo?

Se guardiamo G G M M A A la presentazione ha dei problemi. Confrontate

23 Giugno 1971, o 230671

e 14 Settembre 1973, o 140973.

La data più recente ha il numero più piccolo. Mentre con

4 Luglio 1933, o 040733

e 15 Gennaio 1967, o 150167

la data più recente ha il numero più grande. Chiaramente, quindi, G G M M A A non è un sistema utilizzabile per archiviare delle date.

La soluzione è di sistemare le date nella forma A A M M G G. Questo da alle quattro date esposte prima la forma:

330704, 670115, 710623 e 730914

una consistenza sia come date che come numeri.

Le date sono normalmente memorizzate nella macchina come numeri per essere usate nei calcoli ma sono inserite come stringhe per consentire lo svolgimento di procedure di verifica prima che vengano memorizzate.

Se siamo interessati a calcolare l'incremento dello stipendio, allora dovrebbe essere importante la parte anno e mese del numero che rappresenta la data. Se invece abbiamo un negozio e dobbiamo inviare della pubblicità ai nostri clienti ogni tre mesi, allora può essere importante solo il mese. L'intera stringa di dati è importante in se, ma possiamo vedere che ci possono essere delle ragioni valide per suddividerla.

X$ (A TO B)

Se vogliamo utilizzare parte di una stringa, allora abbiamo bisogno di una istruzione BASIC che faccia questo per noi. Sullo ZX81, l'istruzione è

X$ (A TO B)

Questa ci da dei caratteri di X$ partendo dalla posizione A fino alla posizione B. Per cui:

se X$="STRINGA"
allora X$(3 TO 5) = RIN

Potete verificare come funziona con il seguente programma.

```
   10 REM **SUDDIVISIONE DI
            STRINGHE **
   12 PRINT "INSERIRE LA PAROLA D
 A TAGLIARE"
   14 INPUT M$
   15 PRINT M$
   16 PRINT "INSERIRE INIZIO TAGL
 IO"
   30 INPUT A
   40 PRINT A
   50 PRINT "INSERIRE FINE TAGLIO
 "
   60 INPUT B
   70 PRINT B
   80 LET N$=M$(A TO B)
   90 PRINT "M$(";A;" TO ";B;")="
 ;N$
  100 STOP
```

Programma 11

K Programma 11.

TEST 7

Se A$="1A2B3C4D", cosa producono le seguenti istruzioni?

(a) A$(3 TO 5)
(b) A$ (1 TO 4)
(c) A$ (4 TO 4)

LEFT$, RIGHT$ E MID$

In altri dialetti BASIC troverete

LEFT$(X$,I)
RIGHT$(X$,I)
MID$(X$,I,J)

LEFT$(X$,I) ritorna i caratteri da 1 a I partendo dalla sinistra della stringa X$ così

LEFT$(X$,I) = X$ (1 TO I)

RIGHT$(X$,I) ritorna un numero di caratteri uguale ad I a partire dalla destra della stringa X$ così

RIGHT$ (X$, I) = X$ (LEN X$-I+1 TO LEN X$)

MID$(X$,I,J) ritorna un numero di caratteri uguale a J a partire dal carattere I della stringa X$ così

MID$(X$,I,J) = X$ (1 TO I+J-1)

Suddividere stringhe di lunghezza variabile

Il programma che segue produce tutte le possibili sottostringhe a partire dalla sinistra di qualsiasi parola inserite. Notate che il programma misura la lunghezza della parola che inserite con LEN(X$).

```
  20 PRINT "INSERIRE UNA STRINGA
"
  30 INPUT X$
  40 PRINT X$
  45 PRINT TAB (16);"1234567890"
  50 FOR I=1 TO LEN (X$)
  55 LET A$=X$(1 TO I)
```

LEN (X$) agisce da limite superiore del ciclo.

```
60 PRINT I,A$
70 NEXT I
80 STOP
```

Programma 12

```
RUN
INSERIRE UNA STRINGA
STAMPANTE
                    1234567890
1                   S
2                   ST
3                   STA
4                   STAM
5                   STAMP
6                   STAMPA
7                   STAMPAN
8                   STAMPANT
9                   STAMPANTE
```

K Programma 12.

Esercizio 3

Scrivete un programma che vi consenta di inserire delle parole una alla volta e che visualizzi quelle parole che cominciano con una vocale.

Esercizio 4

Scrivete un programma che cambi la stampa del Programma 12 in:

```
     F
    EF
   DEF
  CDEF
 BCDEF
ABCDEF
```

TEST 8

Scrivete un programma che accetti in input dei numeri telefonici di Roma o Milano nella forma 02 xxxx xxxx e visualizzi il solo numero senza il prefisso. (Ricordate che gli spazi sono caratteri allo stesso modo delle cifre.)

Programma stringhe mediane

Come avrete probabilmente scoperto, sia MID$ che X$ (A TO B) possono tagliare, se ci occorre, le sottostringhe destra e sinistra. In altre parole, esse possono ritornarci ogni possibile sottostringa. Qui di seguito vi è un programma che fa questo. Per prima cosa stampa tutte le sottostringhe di lunghezza 1, quindi tutte quelle di lunghezza 2 e così via finché non arriva a stampare l'intera parola che è la sola sottostringa di lunghezza uguale alla parola stessa!

```
   10 REM **TEST STRINGHE**
   20 PRINT "INSERIRE UNA STRINGA
"
   21 INPUT X$
   22 PRINT X$
   25 FOR J=1 TO LEN (X$)
   26 CLS
   27 PRINT X$;TAB (8);J;" LETTER
E"
   28 PRINT
   30 PRINT "J";TAB (6);":";TAB (
10);"1234567890"
   40 FOR I=1 TO (LEN (X$)-J+1)
   50 LET A$=X$(I TO I+J-1)
   60 PRINT J;TAB (6);":";TAB (10
);A$
   70 NEXT I
   72 PAUSE 100
   75 NEXT J
   80 STOP
```

Produce una pausa prima che il computer visualizzi la successiva tabella

Programma 13

```
RUN
PAROLA    1 LETTERE

J      :    1234567890
1      :    P
1      :    A
1      :    R
1      :    O
1      :    L
1      :    A
PAROLA    2 LETTERE

J      :    1234567890
2      :    PA
2      :    AR
2      :    RO
2      :    OL
2      :    LA
```

```
PAROLA  3 LETTERE

J       :   1234567890
3       :   PAR
3       :   ARO
3       :   ROL
3       :   OLA

PAROLA  4 LETTERE

J       :   1234567890
4       :   PARO
4       :   AROL
4       :   ROLA

PAROLA  5 LETTERE

J           1234567890
5           PAROL
5           AROLA

PAROLA  6 LETTERE

J           1234567890
6           PAROLA
```

K   Programma 13.

5.7 VAL

Avendo trovato un sistema per suddividere le stringhe, ora abbiamo bisogno di un sistema per esaminare quello che otteniamo. Tale sistema è di usare VAL (A$), che guarda al valore numerico di A$.

VAL (A$) sullo ZX81 ci ritorna il valore numerico della stringa A$ purché A$ contenga solo numeri.

Programma dimostrativo di VAL

Il seguente programma ritorna il valore numerico di qualsiasi stringa venga inserita, tramite VAL (N$).

```
 10 REM **LA FUNZIONE VAL **
 20 PRINT "INSERIRE UNA STRINGA
"
 21 INPUT N$
 22 PRINT N$
 25 IF N$="ZZZZ" THEN GOTO 999
 30 LET N=VAL (N$)
 60 PRINT
 70 PRINT N
 80 PRINT
 90 GOTO 20
999 STOP
```

Programma 14

Notate il messaggio di errore. Lo ZX81 non può calcolare VAL (56T) poiché 56T è solo parzialmente numerico.

K Programma 14.

TEST 9

Quali sono i valori delle seguenti espressioni?

(a) VAL (A$) dove A$=54
(b) VAL (B$) dove B$=76XY
(c) VAL (C$) dove C$=A3
(d) VAL (D$) dove D$=—132
(e) VAL (E$1 TO 2)) dove E$=593
(f) VAL (F$(1 TO 1)) dove F=$8AM
(g) VAL (G$(1 TO 2)) dove G$=Z35
(h) VAL (H$(3 TO 3)) dove H.=593
(i) VAL (I$(2 TO 3)) dove I$=8AM
(j) VAL (J$(2 TO 3)) dove J$=Z35

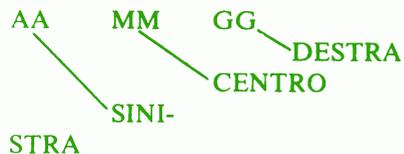
Verifica di stringhe di date

Ora siamo nella condizione per vedere come VAL può essere usato in pratica; in questo caso per verificare l'accuratezza di date inserite nel computer. Questo è tipico di ciò che avviene nei computers continuamente. Noi conosciamo gli errori che avvengono più frequentemente nell'inserire una data così, dove possibile, proviamo ad usare il computer per scoprire gli errori.

Esempio 5

Scriviamo un programma per effettuare la verifica sui 3 campi di una data costituita da una stringa con 6 cifre.

Soluzione

La stringa con la data ha tre campi nella forma:

Noi considereremo soltanto gli anni 1980 e 1981, così:

VAL (D$(1 TO 2)) dovrebbe essere tra 80 e 81,
VAL (D$(3 TO 4)) dovrebbe essere tra 1 e 12,
VAL (D$(5 TO 6)) dovrebbe essere tra 1 e 31.

Questo è un processo abbastanza complesso nel quale per prima cosa dobbiamo decidere sui passi che vanno svolti per la verifica. Questi sono esposti nel seguente diagramma di flusso (Figura 3).

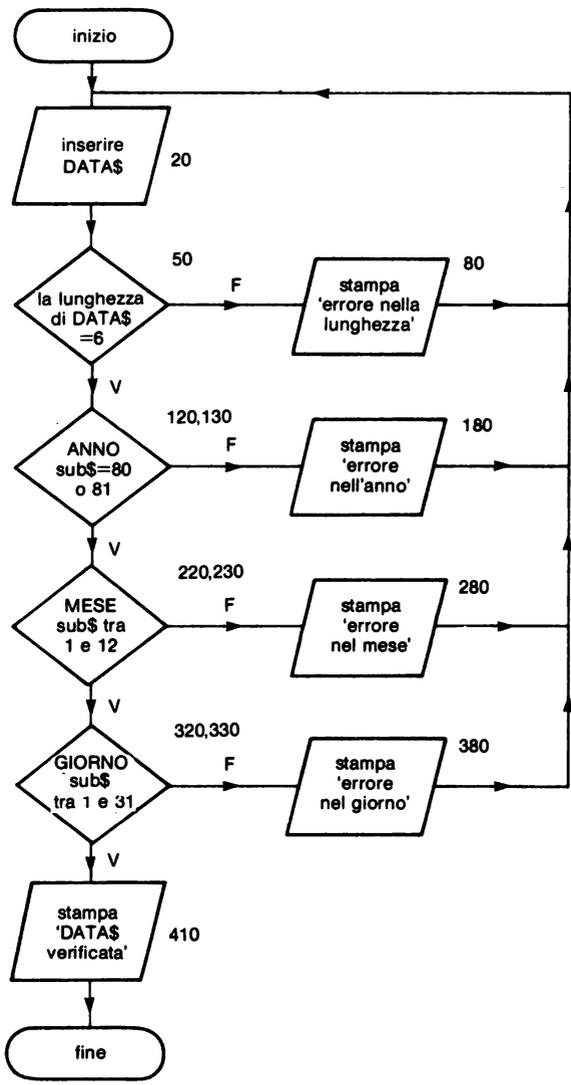



Figura 3 Diagramma di flusso che mostra le quattro verifiche sulla stringa AAMMGG

Il programma è abbastanza chiaro circa il modo in cui vengono effettuate le quattro verifiche. Se qualcuna delle verifiche fallisce, il programma stampa un messaggio di errore e restituisce il controllo alla riga 10. Se non ci sono errori si arriva alla riga 410 dove viene confermato il corretto inserimento della data.

```
  10 REM **VERIFICA DATE**
  15 PRINT
  20 PRINT "PROSSIMA DATA? ";
  21 INPUT D$
  22 PRINT D$
  30 IF D$="ZZZZ" THEN GOTO 900
  50 IF LEN (D$)=6 THEN GOTO 110      verifica lunghezza
  80 PRINT "ERRORE NELLA LUNGHEZ
ZA            DELLA DATA"
  90 GOTO 15
 100 REM ***********************
 110 PRINT "LUNGHEZZA STRINGA CO
RRETTA"
 120 IF VAL (D$(1 TO 2))=80 THEN
 GOTO 210
 130 IF VAL (D$(1 TO 2))=81 THEN      verifica anno
 GOTO 210
 180 PRINT "ERRORE NELL ANNO"
 190 GOTO 15
 200 REM ***********************
 210 PRINT "ANNO CORRETTO"            solo dimostrazione
 220 IF VAL (D$(3 TO 4))<1 THEN
GOTO 280
 230 IF VAL (D$(3 TO 4))<=12 THE      verifica mese
N GOTO 310
 280 PRINT "ERRORE NEL MESE"
 290 GOTO 15
 300 REM ***********************
 310 PRINT "MESE CORRETTO"            solo dimostrazione
 320 IF VAL (D$(5 TO 6))<1 THEN
GOTO 380
 330 IF VAL (D$(5 TO 6))<=31 THE      verifica giorno
N GOTO 410
 380 PRINT "ERRORE NEL GIORNO"
 390          15
 400 REM ***********************
 410 PRINT "LA DATA RIENTRA NEI
VAORI         STABILITI"
 490 GOTO 15
 900 PRINT "FINE VERIFICA DATE"
```

messaggi inclusi a scopo dimostrativo; normalmente non appaiono in una routine di verifica.

Programma 15

Funzionamento tipo

```
RUN
PROSSIMA DATA? 1234567
ERRORE NELLA LUNGHEZZA
DELLA DATA

PROSSIMA DATA? 123456
LUNGHEZZA STRINGA CORRETTA
ERRORE NELL ANNO

PROSSIMA DATA? 803456
LUNGHEZZA STRINGA CORRETTA
ANNO CORRETTO
ERRORE NEL MESE

PROSSIMA DATA? 801256
LUNGHEZZA STRINGA CORRETTA
ANNO CORRETTO
MESE CORRETTO
ERRORE NEL GIORNO

PROSSIMA DATA? 800131
LUNGHEZZA STRINGA CORRETTA
ANNO CORRETTO
MESE CORRETTO
LA DATA RIENTRA NEI VAORI
STABILITI

PROSSIMA DATA? ZZZZ
FINE VERIFICA DATE
```

K Programma 15.

Obiettivi del capitolo 5

Verificate di essere ora in grado di scrivere dei semplici programmi:

Usando LEN (A$) ☐
Usando LET C(I)=C(I)+1 per contare
delle frequenze ☐
Per stampare un diagramma di frequenza ☐
Usando TAB per stampare in colonna ☐
Usando TAB per stampare un diagramma di frequenza
con titoli e scala ☐
Usando X$(A TO B) al posto di LEFT$(X$,I),
RIGHT$(X$,I) e MID$(X$,I,J) ☐
Usando VAL (A$) ☐

Risposte ai TEST ed agli Esercizi

TEST 1

(a) 3; (b) 1; (c) 2 (non 72 - LEN conta il numero di caratteri); (d) 7 (LEN conta i caratteri senza guardare se sono numeri, lettere o spazi).

TEST 2

La vostra risposta dovrebbe essere:

Lunghezza parola	Conteggio	Totale
1	1	1
2	0	
3	111	3
4	11111 111	8
5	1111	4
6	11	2
7	111	3
8	1	1
9	0	
10	0	
11	0	
12	0	
13	0	
14	0	
15	0	

TEST 3

(a) C(4)=1 (non 3 o 4! C(4) ha contato il numero di 3 inseriti)
(b) C(10)=2
(c) C(2)=4
(d) C(1)=3

Esercizio 1

La soluzione appare più avanti nel testo che segue l'esercizio.

TEST 4

```
**
*****
*******
********
***
*
```

TEST 5

```
   10 PRINT "COL 1";TAB (9);"COL
2";TAB (19);"COL 3"
```

TEST 6

```
   10 INPUT A
   20 INPUT B
   30 INPUT C
   40 PRINT TAB (A);"TITOLO";TAB
(B);"TITOLO";TAB (C);"TITOLO"
```

Programma 16

Esercizio 2

Il programma vi chiede di inserire le vocali una per una insieme alla loro frequenza.

```
10 REM **DISTRIBUZIONE DI
         FREQUENZA**
15 REM *PREPARAZIONE DISEGNO*
20 REM **LA LISTA DELLE FRE-
         QUENZE E F(K) **
25 DIM V$(5,1)
28 DIM F(5)
30 FOR K=1 TO 5
35 PRINT "VOCALE N. ";K;" ";
36 INPUT V$(K)
38 PRINT V$(K)
40 PRINT "FREQUENZA ";K;" ";
```

```
  43 INPUT F(K)
  46 PRINT F(K)
  50 NEXT K
  90 CLS
 120 REM ***********************
 200 REM **ROUTINE DI STAMPA   **
 210 PRINT
 212 PRINT "VOCALE";TAB (8);"FRE
Q.";TAB (18);"GRAFICO"
 214 PRINT "====================
============"
 220 FOR X=1 TO 5
 222 PRINT TAB (2);V$(X);TAB (7)
;":";TAB (10);F(X);TAB (14);":";
TAB (16);
 230 FOR Y=1 TO F(X)
 240 PRINT "*";
 250 NEXT Y
 260 PRINT
 280 NEXT X
 290 PRINT "....SCALA....";TAB (
15);"0....5....0....5."
 300 REM ***********************
```

stampa le righe di asterischi

Programma 17

```
VOCALE  FREQ.     GRAFICO
================================
  A    :  9   : *********
  E    :  16  : ****************
  I    :  10  : **********
  O    :  10  : **********
  U    :  6   : ******
....SCALA....  0....5....0....5.
```

K Programma 17.

TEST 7

(a) 2B3; (b) 1A2B; (c) B.
Notate che nel trattamento delle stringhe, tutti i caratteri sono trattati nello stesso modo. Non fa differenza se sono numeri o lettere, essi vengono contati ugualmente.

Esercizio 3

```
  10 REM **IL CARATTERE PIU A
         SINISTRA E UNA
         VOCALE? **
```

```
  20 PRINT "PROSSIMA PAROLA"
  25 INPUT W$
  28 CLS
  30 IF W$="ZZZZ" THEN GOTO 9999
  40 LET L$=W$(1 TO 1)
  50 IF L$="A" THEN GOTO 200
  60 IF L$="E" THEN GOTO 200
  70 IF L$="I" THEN GOTO 200
  80 IF L$="O" THEN GOTO 200
  90 IF L$="U" THEN GOTO 200
 100 GOTO 20
 190 REM ***********************
 200 PRINT
 210 PRINT L$,W$
 220 GOTO 20
 230 REM ***********************
9999 STOP
```

Programma 18

K Programma 18.

Esercizio 4

```
 10 REM **TEST STRINGHE**
 20 PRINT "INSERIRE UNA STRINGA
DI 6 CARAT-TERI"
 25 INPUT X$
 30 PRINT TAB (10);"1234567890"
 40 FOR I=1 TO 6
 50 LET A$=X$(6-I+1 TO 6)
 60 PRINT I;TAB (16-I);A$
 70 NEXT I
 80 STOP
```

Programma 19

RUN

```
INSERIRE UNA STRINGA DI 6 CARAT-
TERI
          1234567890
1                  F
2                 EF
3                DEF
4               CDEF
5              BCDEF
6             ABCDEF
```

K Programma 19.

TEST 8

```
  10 PRINT "PROSSIMO N.RO DI TEL
EFONO"
  20 INPUT N$
  30 PRINT N$
  40 LET A$=N$(4 TO 6)
  50 PRINT A$
  60 GOTO 10
```

Programma 20

K Programma 20.

TEST 9

(a) 54; (b) non VALutabile; (c) non VALutabile; (d) -132; (e) 59; (f) 8; (g) non VALutabile; (h) 3; (i) non VALutabile; (j) 35.

CAPITOLO 6

SOPRATTUTTO SU DADI E GIOCHI

6.1 Numeri casuali

Nella programmazione la funzione che aggiunge un senso di divertimento in un programma è quella che genera numeri casuali. Questa funzione è il cuore di molti dei programmi di giochi e di simulazioni che sono ora disponibili per i microcomputers.

Voi avrete incontrato i numeri casuali nel fare dei giochi; giochi che richiedono il lancio di una moneta o di un dado, o estrarre dei numeri da un cappello. Altri giochi che si basano su dei numeri casuali sono ad esempio il lotto ed i giochi da casinò. Nonostante tutti noi abbiamo un'idea intuitiva di cosa intendiamo per sequenza di numeri casuali, è abbastanza difficoltoso definire tale idea chiaramente. Diamo uno sguardo ad alcune sequenze di numeri per provare a chiarire questa idea.

Qui di seguito vi sono tre esperimenti immaginari, ciascuno dei quali coinvolge il lancio di un dado a sei lati per quindici volte. Immaginate che nel primo esperimento i punteggi del dado (il valore della faccia rivolta verso l'alto) abbiano i valori indicati nella sequenza A visibile in Figura 1, che il secondo esperimento generi i numeri esposti nella sequenza B ed il terzo esperimento ci dia i numeri mostrati nella sequenza C.

Sequenza A
5,1,2,4,6,3,2,1,6,3,5,4,3,4,2
Sequenza B
6,6,6,6,6,6,6,6,6,6,6,6,6,6,6
Sequenza C
1,2,3,4,5,6,1,2,3,4,5,6,1,2,3

Figura 1 Sequenze casuali?

Possiamo essere abbastanza sicuri che la sequenza A rappresenti una tipica sequenza di numeri prodotta dal lancio di un dado per quindici volte. Questa sequenza di numeri non mostra nessun modello predefinito ne ripetizioni e l'uscita di un qualsiasi numero sembrerebbe essere ugualmente probabile. E non siamo sorpresi se appaiono delle sub-sequenze all'interno della sequenza principale. Per contrasto, invece, la sequenza B è abbastanza irragionevole. Noi certamente non ci aspetteremmo di avere quindici sei con quindici lanci consecutivi del dado. Saremmo piuttosto sorpresi se ciò avvenisse e penseremmo che il dado sia truccato. Intuitivamente, noi saremmo pronti ad accettare che la sequenza A sia capitata 'per caso' ma non saremmo pronti ad ammettere che sia così anche per la sequenza B.

Un'altra caratteristica delle sequenze di numeri casuali che conosciamo per esperienza è che, nelle lunghe sequenze, le combinazioni sfavorevoli si compensano. Quel che vogliamo dire è che dopo, diciamo, un centinaio di lanci, noi dovremmo

aspettarci mediamente circa sedici uno, circa sedici due, sedici tre e così via. In altre parole, durante una lunga sequenza ci aspettiamo che si applichi la 'legge delle probabilità'. Se ora consideriamo la sequenza C con il suo evidente modello '...6,1,2,3,4,5,6,1...' continuato per un centinaio di lanci la regola delle uguali probabilità nel lungo termine sarebbe soddisfatta. Ma ancora una volta questa sequenza non ci sarebbe intuitivamente accettabile come casuale perché noi non potremmo aspettarci che questo modello sequenziale persista per oltre un centinaio di lanci solo per caso.

Questi concetti di 'media statistica' durante le lunghe sequenze e di 'ragionevolezza' delle serie di numeri nella sequenza sono intuitivamente acquisibili dai giochi basati sul caso. Ci sono delle tecniche statistiche per provare queste due caratteristiche di una sequenza di numeri casuali, ma non ci interesseremo qui di queste tecniche.

Il computer è una macchina molto precisa. Non sarete, perciò, sorpresi nel sapere che per ottenere una sequenza di numeri casuali nella macchina devono essere previste delle prestazioni abbastanza speciali. Per il nostro uso, comunque, noi supporremo che nella memoria della macchina sia stata memorizzata una tabella di numeri casuali. La sequenza di numeri è molto lunga e la generazione dei numeri dovrebbe continuare per un lungo periodo di tempo prima che la ripetizione della sequenza diventi evidente. Per ottenere una diversa sequenza di numeri casuali da una esecuzione di programma ad un'altra , tutto ciò che la macchina deve fare e di iniziare la lettura di questa tabella di numeri casuali da punti diversi. Questo punto di partenza è spesso indicato come il 'seed' (principio) e noi parliamo delle sequenze di numeri casuali come inizianti da differenti seeds. Poiché il computer deve 'inventare' le sequenze di numeri casuali, i numeri prodotti sono usualmente indicati come numeri pseudo-casuali.

6.2 La funzione RND

Se fate girare il seguente programma sarete in grado di vedere l'effetto di RND.

```
10 FOR I=1 TO 10
30 LET B=RND
40 PRINT B
50 NEXT I
```

Programma 1 RND

Facendo girare il programma si ottengono numeri come:

```
0.6658783
0.94125366
0.59408569
```

```
0.55688477
0.76686096
0.51483154
0.61291504
0.96907044
0.68031311
.023834229
```

K Programma 1.

Questi certamente appaiono come numeri casuali e, se fate girare ancora il programma, otterrete una lista completamente differente. In realtà il computer ha una lista di 65536 numeri mescolati e RND si inserisce in questa lista e legge tanti numeri quanti ne avete chiesti.

Il fatto che i numeri casuali sono disponibili in una sequenza fissa ci consente di ripetere se vogliamo una data sequenza di numeri casuali. Sullo ZX81 noi otteniamo ciò tramite la keyword RAND, per esempio, RAND 25 farà iniziare la lettura dei numeri della sequenza dal 25esimo numero. Provate il seguente programma per convincervi che RAND stabilisce il punto di partenza della sequenza.

```
20 RAND 7
40 FOR I=1 TO 10
50 LET B=RND
60 PRINT B
70 NEXT I
```

Programma 2

K Programma 2. Fatelo girare diverse volte e notate la sequenza.

RND

Quel che dimostrano le prove qui sopra e che

RND

fornirà dei numeri casuali compresi tra 0 e 1 con 0 incluso e 1 escluso.

Come possiamo estendere la gamma per ottenere altri numeri casuali? Semplicemente moltiplicando RND con un'altro numero. Così:

RND da un numero casuale tra 0 e 1;

6*RND da un numero casuale tra 0 e 6;

e 52*RND da un numero casuale tra 0 e 52;
ecc.

Potete pensare a RND come ad un 'fattore di conversione' il quale cambia di continuo. Il seguente programma mette in pratica questa idea.

```
   10 REM **RND COME FATTORE
            DI CONVERSIONE **
   20 PRINT " I";TAB (8);"1*RND";
TAB (20);"6*RND"
   30 PRINT "---";TAB (8);"-----"
;TAB (20);"-----"
   40 FOR I=1 TO 10
   50 LET B=RND
   60 LET C=6*B
   70 PRINT I;TAB (8);B;TAB (20);
C
   80 NEXT I
   90 STOP
```

Programma 3 RND come fattore di conversione

Fate girare il programma con la riga 70 come LET C=6*B

```
 I RUN    1*RND        6*RND
---       -----        -----
1         0.6658783    3.9952698
2         0.94125366   5.647522
3         0.59408569   3.5645142
4         0.55688477   3.3413086
5         0.76686096   4.6011658
6         0.51483154   3.0889893
7         0.61291504   3.6774902
8         0.96907044   5.8144226
9         0.68031311   4.0818787
10        .023834229   0.14300537
```

Provate a girare il programma con la riga 70 come LET C=52*B più il cambio del titolo nella riga 20.

RUN

```
 I         1*RND         52*RND
---        -----         ------
1          0.69326782    36.049927
2          0.99543762    51.762756
3          0.65782166    34.206726
4          0.33700562    17.524292
5          0.27616882    14.360779
6          0.71348572    37.101257
7          0.51174927    26.610962
8          0.38174439    19.850708
9          0.63153076    32.8396
10         0.36521912    18.991394
```

K Programma 3.

TEST 1

Scrivete un programma per stampare 6 numeri casuali compresi tra 0 e 5.999999.

La funzione RND+1

Se osservate di nuovo la stampa del Programma 3 con 6*RND, i numeri erano:

```
3.9952698
5.647522
3.5645142
3.3413086
4.6011658
3.0889893
3.6774902
5.8144226
4.0818787
0.14300537
```

Date ora uno sguardo ai numeri prima del punto decimale. Essi sono:

3,5,3,3,4,3,3,5,4,0

vale a dire componenti della serie

(0,1,2,3,4,5)

Ma, se noi avessimo lanciato il dado avremmo dovuto generare dei numeri della serie (1,2,3,4,5,6). Tutto ciò che dobbiamo fare, è di aggiungere 1 a ciascuno dei componenti della prima serie per ottenere la seconda.

Ora, nei giochi e richiesto di frequente il lancio di un dado (risultati 1,2,3,4,5,6) o l'uso di un mazzo di carte (52 risultati) così noi siamo particolarmente interessati alle funzioni:

6*RND+1 e 52*RND+1

Il seguente programma ci consente di esaminare la funzione 6*RND+1.

```
   10 REM **RND COME FATTORE
           DI CONVERSIONE **
   20 PRINT " I";TAB (8);"1*RND";
TAB (20);"6*RND+1"
   30 PRINT "---";TAB (8);"-----"
;TAB (20);"-------"
   40 FOR I=1 TO 10
   50 LET B=RND
   60 LET C=6*B+1
   65 REM **NOTATE IL +1 **
   70 PRINT I;TAB (8);B;TAB (20);
C
   80 NEXT I
   90 STOP
```

Nota + 1

*Programma 4 6*RND+1*

```
 I RUN    1*RND        6*RND+1
---       -----        -------
1         0.16993713   2.0196228
2         0.74623108   5.4773865
3         0.96762085   6.8057251
4         0.57159424   4.4295654
5         0.87005615   6.2203369
6         0.25434876   2.5260925
7         .07699585    1.4619751
8         0.77574158   5.6544495
9         0.18086243   2.0851746
10        0.56561279   4.3936768
```

K Programma 4.

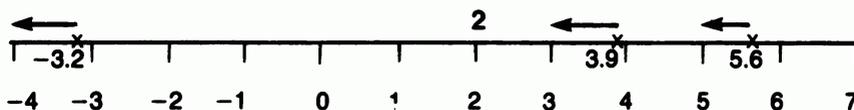
La funzione INT

Se ora osservate la prima cifra della terza colonna risultante dal funzionamento del Programma 4, vedrete che noi abbiamo prodotto i numeri casuali di cui avevamo bisogno. La colonna 3 ha i numeri da 1 a 6; se cambiate la riga 70 in LET C=52*B+1 ,modificate il titolo della colonna nella riga 20, e fate girare di nuovo il Programma 4, troverete che la colonna 3 ha numeri da 1 a 52.

Ma che cosa ne facciamo di tutti i numeri non necessari che si trovano dopo il punto decimale? Bene, noi abbiamo una funzione per liberarci di questi numeri: la funzione INT.

L'effetto di INT (X) è di ritornare la parte intera del numero X, vale a dire il numero intero più grande che non sia maggiore di X. L'effetto di INT è di ritornare il successivo più alto numero intero:

INT (5.6) = 5
INT (3.9) = 3
INT (-3.2) = -4
INT (2) = 2

Se INT (-3.2)=-4 vi sorprende osservate la linea dei numeri in basso e ricordate che INT ritorna sempre il successivo numero intero e non 'arrotonda' i numeri.

TEST 2

Quali sono i valori delle seguenti espressioni?
(a) INT (4.5)
(b) INT (9.1)
(c) INT (-2.5)
(d) INT (-0.99)
(e) INT (1.01)

Ora, con INT, possiamo finalmente generare i numeri interi da 1 a 6 e da 1 a 52 per usarli come dadi o carte. Tutto ciò che occorre è

```
INT (6*RND+1)
e INT (52*RND+1)
```

Il seguente programma visualizza i valori di queste due funzioni:

```
  10 REM **LA FUNZIONE INT**
  30 PRINT " I";TAB (5);"RND*1";
TAB (12);"INT(6*---";TAB (22);"I
NT(52*--"
  35 PRINT TAB (12);"---RND+1)";
TAB (22);"---RND+1)"
  40 PRINT "---";TAB (5);"-----"
;TAB (12);"---------";TAB (22);"
---------"
  45 PRINT
  50 FOR I=1 TO 10
  60 LET B=RND
  70 LET C=INT (6*B+1)
  80 LET D=INT (52*B+1)
  90 PRINT I;TAB (4);B;TAB (18);
C;TAB (25);D
 100 NEXT I
 110 STOP
```

Programma 5 La funzione INT per dare numeri interi

RUN

```
 I    RND*1    INT(6*---  INT(52*--
               ---RND+1)  ---RND+1)
---  -----    ---------  ---------

1   0.36521912     3        19
2   0.39215088     3        21
3   0.41200256     3        22
4   0.90086365     6        47
5   0.56488037     4        30
6   0.36651611     3        20
7   0.48942566     3        26
8   0.70750427     5        37
9   .063140869     1        4
10  0.7366333      5        39
```

K Programma 5.

6.3 Numeri casuali poscritti

Il seguente programma simula il lancio di un dado per 100 volte:

```
 10 REM *100 LANCI DI UN DADO*
 40 FOR I=1 TO 10
 50 FOR J=1 TO 10
 60 LET X=INT (6*RND+1)
 70 PRINT X;" ";
 80 NEXT J
100 PRINT
110 NEXT I
120 STOP
```

Programma 6 Lancio di un dado

```
RUN
4  3  3  3  6  4  3  3  5  1
5  2  4  6  2  5  5  2  6  5
1  6  6  6  1  6  6  6  6  5
3  2  2  3  4  6  6  2  5  5
5  4  1  3  4  3  1  5  6  1
1  1  1  6  5  1  2  5  1  1
5  3  3  6  4  6  2  1  3  4
5  5  4  2  6  5  4  2  2  3
5  6  6  1  1  3  4  3  6  3
2  1  6  4  3  6  6  2  5  4
```

Programma 6.

TEST 3

Il programma:

```
10 REM **TEST 3**
20 FOR X=-3.8 TO -1.8 STEP .2
30 LET Y=INT X
40 PRINT X,Y
50 NEXT X
60 STOP
```

Programma 7

stampa 10 paia di numeri. Quali sono?

TEST 4

Il programma:

```
10 REM **TEST 4**
20 FOR X=1.6 TO 3.4 STEP .2
30 LET Y=INT X
40 PRINT X,Y
50 NEXT X
60 STOP
```

Programma 8

stampa 9 paia di numeri. Quali sono?

6.4 Due esempi

Questa sezione è composta da due lunghi esempi. Vi suggeriamo di provare a trattarli prima come esercizi e quindi confrontare le vostre soluzioni con le nostre.

Esempio 1

Scriviamo un programma per simulare il lancio di una moneta per 100 volte. Contare il numero di volte che la moneta cade su testa e su croce e visualizzare il risultato.

Soluzione

Il nucleo della soluzione è un generatore di numeri casuali che produce un 1 o un 2. Noi useremo 1 per rappresentare testa e 2 per rappresentare croce. Usando questo approccio, un'algoritmo descrittivo per la soluzione del problema è:

1. Inizio.
2. Contatori del totale testa e del totale croce a zero.
3. Inizio del ciclo di conteggio.
4. Genera i due valori 1 e 2 casualmente.
5. Se il numero casuale è uguale a 2 allora va all'istruzione 8 altrimenti continua con l'istruzione 6.
6. Aggiunge 1 al totale di testa.
7. Va all'istruzione 9.
8. Aggiunge 1 al totale di croce.
9. Aggiunge 1 al contatore del ciclo.

10. Se il contatore di ciclo è < =100 allora va all'istruzione 4 altrimenti continua con l'istruzione 11.
11. Visualizza il totale teste ed il totale croci.
12. Fine.

Figura 2 Soluzione descrittiva del lancio di una moneta

Alternativamente, potreste preferire una descrizione del problema sotto forma di diagramma di flusso:

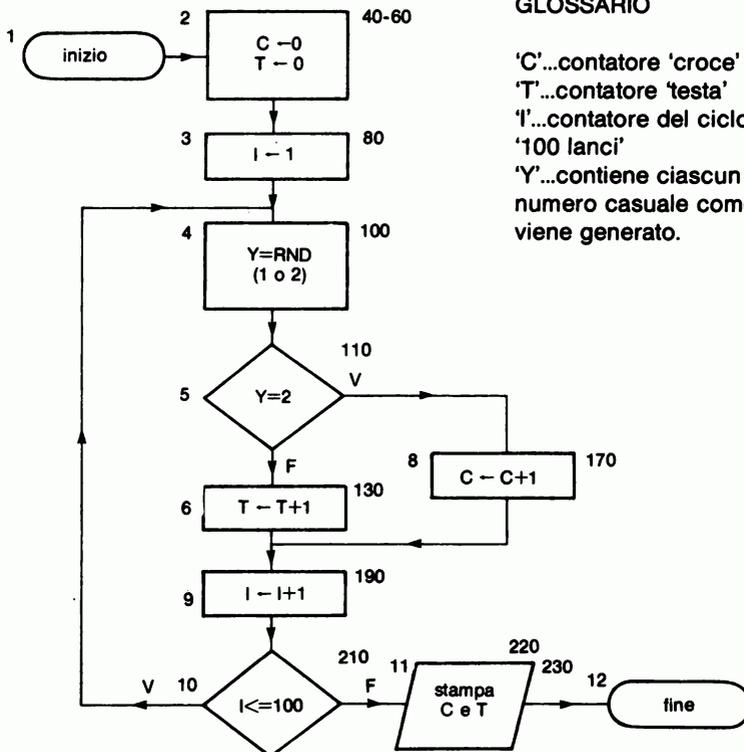



Figura 3 Diagramma di flusso per il lancio della moneta

E, finalmente, il programma:

```
 10 REM **LANCIO DI UNA MONETA
         PER 100 VOLTE **
 40 LET C=0
 60 LET T=0
 80 LET I=1
100 LET Y=INT (2*RND+1)
110 IF Y=2 THEN GOTO 170
130 LET T=T+1
150 GOTO 190
170 LET C=C+1
190 LET I=I+1
210 IF I<=100 THEN GOTO 100
220 PRINT "CROCI","TESTE"
230 PRINT C,T
240 STOP
```

Programma 9 Lancio di una moneta

Tre funzionamenti tipo:

```
RUN
CROCI           TESTE
59              41

CROCI           TESTE
50              50

CROCI           TESTE
49              51
```

K Programma 9.

Esempio 2

Scriviamo un programma per simulare il lancio di due monete per cento volte. Contare il numero di volte che le monete cadono su croce-croce (CC), testa-testa (TT), e croce-testa o testa-croce (CT o TC) e visualizzare il risultato.

Soluzione

Per i punteggi usiamo le stesse regole di prima: 1 è testa e 2 è croce, ma ora lanciamo due monete. Memorizziamo i punteggi della prima moneta in C1 ed i punteggi della seconda moneta in C2. Quindi sommiamo C1 e C2 per dare il punteggio totale per

quel lancio:

S=C1+C2

S può essere 2, 3, 4:

	uscite	punteggio
	TT	1+1=2
TC o CT	1+2=2+1=3	
CC	2+2=4	

Quindi contiamo quanti 2, quanti 3 e quanti 4 otteniamo.

Contatore dei 2 T2
Contatore dei 3 M1 (T e C)
Contatore dei 4 H2

Il diagramma di flusso della soluzione è:

GLOSSARIO DEI SIMBOLI

H2 ...totale per CC
T2 ...totale per TT
M1 ...totale per CT o TC
C1 ...1 o 2 casuale per la moneta 1
C2 ...1 o 2 casuale per la mometa 2
S ...somma dei punteggi delle monete

inizio
40-80
H2 ← 0
T2 ← 0
M1 ← 0
100,310
I ← 1
I ← I+1
I>100
T
F
330,340
stampa
H2, T2, M1
fine
110,130
C1 & C2
=RND (1 o 2)
150
S=C1+C2
170
S=4
T
F
190
S=2
T
F
210
M1 ← M1+1
250
T2 ← T2+1
290
H2 ← H2+1

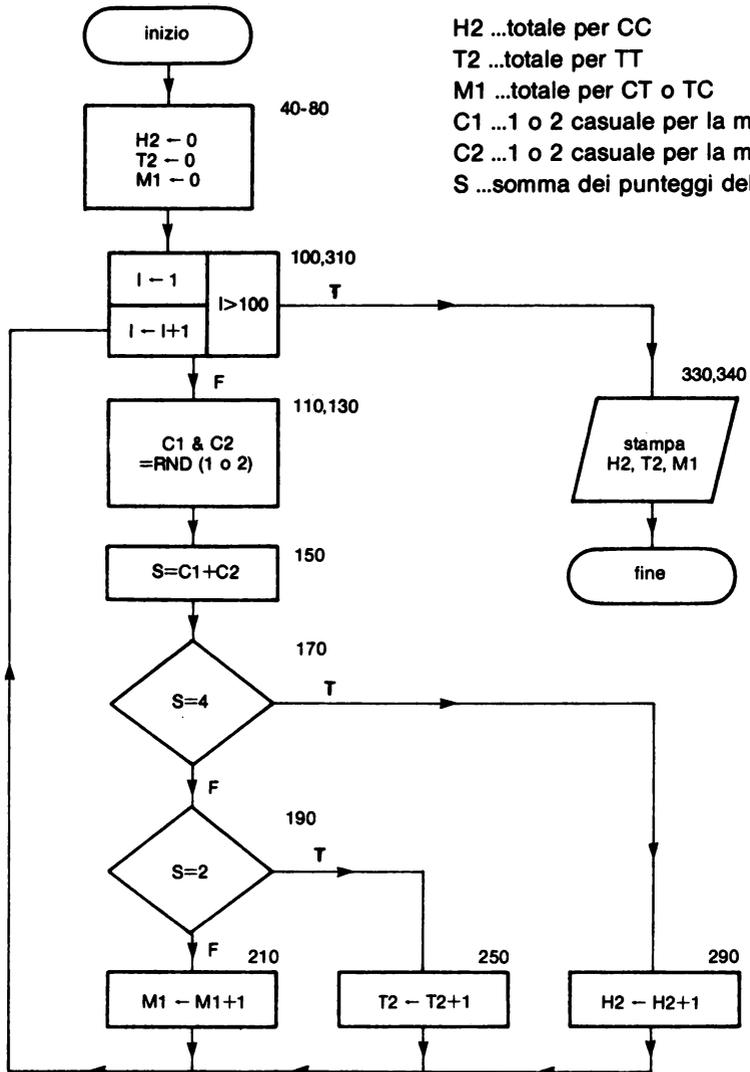



Figura 4 Diagramma di flusso per il lancio di due monete

Ed il programma è:

```
 10 REM **LANCIO DI 2 MONETE
          PER 100 VOLTE **
 40 LET H2=0
 60 LET T2=0
 80 LET M1=0
100 FOR I=1 TO 100
110 LET C1=INT (2*RND+1)
130 LET C2=INT (2*RND+1)
150 LET S=C1+C2
170 IF S=4 THEN GOTO 290
190 IF S=2 THEN GOTO 250
210 LET M1=M1+1
230 GOTO 310
250 LET T2=T2+1
270 GOTO 310
290 LET H2=H2+1
310 NEXT I
330 PRINT "TT";TAB (10);"CT";TA
B (20);"CC"
340 PRINT T2;TAB (10);M1;TAB (2
0);H2
350 STOP
```

Programma 10 Lancio di due monete

Qui ci sono alcuni risultati tipo:

```
RUN

TT        CT        CC
20        61        19

RUN

TT        CT        CC
25        53        22

RUN

TT        CT        CC
31        38        31

RUN

TT        CT        CC
30        51        19
```

K Programma 10.

6.5 Tenere i punteggi

Avrete notato che abbiamo usato alcuni nomi di variabili ineleganti, come T2, H2 e M1. Probabilmente avrete pensato, 'Perché non sono state usate le liste di variabili? Non possono essere altrettanto utili qui come lo sono state nelle tabelle di frequenze?' Infatti possono essere usate, così proviamo ad usare una lista-punteggi, S(I), per il lancio della moneta. Quindi diciamo:

Se il punteggio è 2, aggiungi 1 al numero in S(2)
Se il punteggio è 3, aggiungi 1 al numero in S(3)
Se il punteggio è 4, aggiungi 1 al numero in S(4)

e più in generale.
Se il punteggio è s aggiungi 1 al numero in S(S)

Applicazione al lancio di due monete

Se torniamo indietro all'Esempio 2, possiamo riutilizzare la struttura base, aggiungendo il nostro nuovo sistema di punteggio:

```
160 LET S(S)=S(S)+1
```

Il programma quindi diventa:

```
 10 REM **LANCIO DI 2 MONETE
         PER 100 VOLTE **
 35 DIM S(4)
 40 LET S(4)=0
 60 LET S(3)=0
 80 LET S(2)=0
100 FOR I=1 TO 100
110 LET C1=INT (2*RND+1)
130 LET C2=INT (2*RND+1)
150 LET S=C1+C2
160 LET S(S)=S(S)+1
310 NEXT I
330 PRINT "TT";TAB (10);"CT";TA
B (20);"CC"
340 PRINT S(2);TAB (10);S(3);TA
B (20);S(4)█ 350 STOP
```

Programma 11

```
RUN
TT          CT          CC
25          53          22

TT          CT          CC
31          38          31

TT          CT          CC
30          51          19

TT          CT          CC
25          49          26
```

K Programma 11.

Liste-punteggi per dadi

La lista-punteggi per il lancio di un dado sarebbe:

S(1),S(2),S(3),...,S(6)

e per il lancio di due dadi:

S(2),S(3),S(4),...,S(12).

Esercizio 1

Scrivete un programma per simulare il lancio di un dado per 100 volte, contare e visualizzare il numero di volte in cui esce ciascun punteggio.

Esercizio 2

Modificate il programma scritto per l'Esercizio 1 per simulare il lancio di due dadi per 100 volte.

Esercizio 3

Scrivete un programma per visualizzare i dati ottenuti dal programma dell'Esercizio 2, nella forma di un diagramma di frequenza.

6.6 Concatenazione

Dopo avere descritto nell'Unità 5 il modo di suddividere le stringhe, ora spenderemo un po' di tempo sul come rimettere insieme (o concatenare) dei pezzi di stringhe. Il Programma 12 ci mostra cosa avviene.

```
20 PRINT "PRIMA STRINGA"
25 INPUT A$
30 PRINT A$
40 PRINT "SECONDA STRINGA"
45 INPUT B$
50 PRINT B$
60 PRINT A$+B$
70 STOP
```

Programma 12 Concatenazione

```
RUN
PRIMA STRINGA
MICRO
SECONDA STRINGA
COMPUTER
MICROCOMPUTER

RUN
PRIMA STRINGA
CONCATE
SECONDA STRINGA
NAZIONE
CONCATENAZIONE
```

TEST 5

Scrivete un programma per inserire una parola inglese e visualizzare il suo plurale tenendo presente che la maggior parte delle parole inglesi richiede soltanto l'aggiunta di s per avere il plurale.

Il Programma 13 mostra in che modo possiamo usare la concatenazione per costruire una stringa da una lista di simboli. Abbiamo memorizzato le lettere in A$ alla riga 30; nel ciclo 70-100 aggiungiamo una nuova lettera alla stringa ad ogni passaggio.

```
10 REM **CONCATENAZIONI**
20 REM **COSTRUIRE ARCHIVIO
      SIMBOLI **
30 LET A$="ABCDEFGHIJ"
40 REM ***********************
50 LET C$=""
60 REM **STRINGA C$ VUOTA **
70 FOR J=1 TO 10
80 LET C$=C$+A$(J)
130 PRINT J,C$
140 NEXT J
150 STOP
```

Programma 13

RUN

```
1           A
2           AB
3           ABC
4           ABCD
5           ABCDE
6           ABCDEF
7           ABCDEFG
8           ABCDEFGH
9           ABCDEFGHI
10          ABCDEFGHIJ
```

K Programma 13.

Questo processo è di grande utilità nell'analisi di testi, ma lo useremo per codici e giochi.

6.7 STR$

Questa funzione ha l'effetto inverso alla funzione VAL. La funzione VAL dà il valore numerico di una stringa, e la funzione STR 9$ trasforma un numero in una stringa di caratteri.

STR$(X) dà la rappresentazione in stringa del valore di X.

Stampare STR$

STR$(N) appare molto simile ad N stesso, come mostra il seguente programma:

```
10 REM **LA FUNZIONE STR$ **
20 PRINT "PROSSIMO NUMERO?";
25 INPUT N
30 PRINT N
40 PRINT "01234567890123456789
0"
50 PRINT N,STR$ (N)
60 STOP
```

Programma 14

RUN

```
PROSSIMO NUMERO?17
01234567890123456790
17                 17
```

RUN

```
PROSSIMO NUMERO?-17
01234567890123456790
-17                -17
```

RUN

```
PROSSIMO NUMERO?99.34
01234567890123456790
99.34              99.34
```

RUN

```
PROSSIMO NUMERO?-99.34
01234567890123456790
-99.34             -99.34
```

K Programma 14.

Tuttavia, in ciascun esempio la seconda cifra colorata è trattata come stringa.

Nel prossimo programma (Programma 15) facciamo uso del fatto che, per esempio, STR$ tratta 8 come una stringa, così che possiamo aggiungere il carattere 8 (a differenza del suo valore) alla fine della stringa.

```
10 REM ************************
20 LET C$=""
30 REM **STRINGA C$ VUOTA **
40 FOR J=1 TO 10
50 LET C$=C$+STR$ (J)
60 PRINT J,C$
70 NEXT J
80 STOP
```

Programma 15

Stampa su alcuni microcomputers:

```
RUN
1           1
2           1 2
3           1 2 3
4           1 2 3 4
5           1 2 3 4 5
6           1 2 3 4 5 6
7           1 2 3 4 5 6 7
8           1 2 3 4 5 6 7 8
9           1 2 3 4 5 6 7 8 9
10          1 2 3 4 5 6 7 8 9 10
```

Stampa sullo ZX81:

```
RUN
1           1
2           12
3           123
4           1234
5           12345
6           123456
7           1234567
8           12345678
9           123456789
10          12345678910
```

Quindi sullo ZX81 possiamo unire i caratteri in posizioni adiacenti.

K Programma 15.

Esercizio 4

Scrivete un programma per inserire una parola dalla tastiera, codificare ciascuna lettera come un numero, e stampare i codici come una sequenza di numeri.

Indicazioni se necessarie: costruire una lista-archivio come nel Programma 13 ma con l'intero alfabeto. Ricordate l'istruzione DIM. Prendete ciascuna lettera della parola e confrontatela con gli elementi della lista-archivio. Quando viene trovata nella lista, aggiungete il valore indice, sotto forma di stringa, alla stringa dei codici.

Esercizio 5

Scrivete un programma per generare 20 parole di 3 lettere casuali. (È interessante vedere quante volte occorre far girare questo programma per produrre una parola che abbia un significato.)

Obiettivi del capitolo 6

Quando avete finito questa unità, verificate di essere in grado di:

Usare RND per generare numeri casuali compresi tra 0 e 1. ☐

Usare INT ed RND per generare numeri casuali interi compresi tra 0 ed un dato valore intero N. ☐

Simulare il lancio di una moneta. ☐

Simulare il lancio di due monete. ☐

Simulare il lancio di un dado. ☐

Simulare il lancio di due dadi. ☐

Usare liste di punteggi. ☐

Concatenare stringhe. ☐

Usare STR$(X). ☐

Risposte ai TEST ed agli Esercizi

TEST 1

```
10 FOR I=1 TO 6
20 LET N=6*RND
30 PRINT N
40 NEXT I
50 STOP
```

Programma 16

TEST 2

(a) 4; (b) 9; (c) -3; (d) -1; (e) 1.

TEST 3

RUN

-3.8	-4
-3.6	-4
-3.4	-4
-3.2	-4
-3	-3
-2.8	-3
-2.6	-3
-2.4	-3
-2.2	-3
-2	-2

TEST 4

RUN

1.6	1
1.8	1
2	2
2.2	2
2.4	2
2.6	2
2.8	2
3	3
3.2	3

Esercizio 1

Algoritmo descrittivo per il lancio di 1 dado

1. Inizio.
2. Locazioni dei totali dei 6 punteggi a zero.
3. Inizio del ciclo 100 lanci.
4. Genera un punteggio casuale nella serie (1,2,3,4,5,6).
5. Incrementa il totale collegato a questo punteggio.
6. Se il contatore di ciclo < =100 va all'istruzione 4 altrimenti continua con l'istruzione 7.
7. Visualizza i punteggi ed il totale per ciascun valore di punteggio.
8. Fine.

```
 10 REM **LANIO DI UN DADO
        PER 100 VOLTE **
 50 DIM S(6)
 70 FOR J=1 TO 6
 80 LET S(J)=0
 90 NEXT J
110 FOR I=1 TO 100
120 LET S=INT (6*RND+1)
130 LET S(S)=S(S)+1
150 NEXT I
170 PRINT
180 PRINT "PUNTEGGIO","FREQUENZ
A"
200 FOR K=1 TO 6
210 PRINT K,S(K)
220 NEXT K
230 STOP
```

Programma 17

RUN

```
PUNTEGGIO          FREQUENZA
1                  12
2                  12
3                  17
4                  18
5                  17
6                  24
```

RUN

```
PUNTEGGIO          FREQUENZA
1                  15
2                  18
3                  17
4                  18
5                  14
6                  18
```

RUN

```
PUNTEGGIO          FREQUENZA
1                  12
2                  22
3                  17
4                  14
5                  17
6                  18
```

RUN

```
PUNTEGGIO          FREQUENZA
1                  19
2                  16
3                  20
4                  10
5                  16
6                  19
```

K programma 17.

Esercizio 2

```
  10 REM **LANIO DI 2 DADI
          PER 100 VOLTE **
  50 DIM S(12)
  70 FOR J=2 TO 12
  80 LET S(J)=0
  90 NEXT J
 110 FOR I=1 TO 100
 120 LET S1=INT (6*RND+1)
 125 LET S2=INT (6*RND+1)
 130 LET S=S1+S2
 135 LET S(S)=S(S)+1
 150 NEXT I
 170 PRINT
 180 PRINT "PUNTEGGIO","FREQUENZ
A"
```

```
200 FOR K=2 TO 12
210 PRINT K,S(K)
220 NEXT K
230 STOP
```

Programma 18

RUN

```
PUNTEGGIO          FREQUENZA
2                  2
3                  7
4                  11
5                  5
6                  14
7                  16
8                  21
9                  9
10                 8
11                 2
12                 5
```

K Programma 18.

E per 1000 lanci? Bene, semplicemente cambiate la linea 110 in 110 FOR i=1 TO 1000 ,ed otterrete un risultato del tipo:

RUN

```
PUNTEGGIO          FREQUENZA
2                  23
3                  65
4                  87
5                  99
6                  149
7                  174
8                  131
9                  111
10                 88
11                 49
12                 24
```

Suggerimenti per ulteriori programmi

1. Come si può stampare un diagramma di frequenza per questi dati, se numeri come 147, 172 ... dovrebbero essere stampati oltre la fine della linea?
2. Abbiamo bisogno di una routine universale di riduzione in scala che aggiusti le differenti lunghezze delle linee, ma che nello stesso tempo faccia uso di tutta la larghezza dello schermo.

Esercizio 3

```
 10 REM **LANCIO DI 2 DADI PER
          100 VOLTE PIU DIA-
          GRAMMA DI FREQUENZA *
 50 DIM S(12)
 70 FOR J=2 TO 12
 80 LET S(J)=0
 90 NEXT J
110 FOR I=1 TO 100
120 LET S1=INT (6*RND+1)
125 LET S2=INT (6*RND+1)
130 LET S=S1+S2
135 LET S(S)=S(S)+1
150 NEXT I
160 PRINT
170 PRINT "DIAGRAMMA DI FREQUEN
ZA"
180 PRINT
190 FOR K=2 TO 12
200 PRINT K;TAB (5);S(K);TAB (1
0);
210 IF S(K)=0 THEN GOTO 270
230 FOR L=1 TO S(K)
240 PRINT "*";
250 NEXT L
270 PRINT
280 NEXT K
300 STOP
```

Programma 19

```
RUN

DIAGRAMMA DI FREQUENZA

2    4    ****
3    4    ****
4    4    ****
5    11   ***********
6    16   ****************
7    20   ********************
8    13   *************
```

```
9      11     ***********
10     8      ********
11     6      ******
12     3      ***
```

K   Programma 19.

TEST 5

```
10 REM **PLURALE INGLESE**
20 LET A$="S"
30 PRINT "INSERIRE LA PAROLA"
35 INPUT B$
40 PRINT B$
50 PRINT B$+A$
60 STOP
```

Programma 20

Esercizio 4

```
 10 REM **PROGRAMMA CODICI**
 30 LET C$=""
 40 DIM A$(26)
 50 LET A$="ABCDEFGHIJKLMNOPQRS
TUVWXYZ"
 70 PRINT "PROSSIMA PAROLA DA C
ODIFICARE?"
 75 INPUT W$
 80 PRINT W$
 90 LET L=LEN (W$)
100 FOR J=1 TO L
120 LET I=1
130 IF W$(J)=A$(I) THEN
GOTO 170
150 LET I=I+1
160 GOTO 130
170 LET C$=C$+STR$ (I)+" "
180 NEXT J
190 PRINT
200 PRINT C$
210 STOP
```

realizzazione archivio (righe 40–50)

confronta ciascuna lettera della parola (W$) con ciascuna lettera dell'archivio fino a che non lo trova, (righe 90–150)

quindi aggiunge l'indice in forma di stringa codici. (riga 170)

RUN

```
PROSSIMA PAROLA DA CODIFICARE?
COMPUTER

3 15 13 16 21 20 5 18
```

RUN

```
PROSSIMA PAROLA DA CODIFICARE?
PROESSIONE

16 18 15 5 19 19 9 15 14 5
```

K Programma 21.

Esercizio 5

```
  10 REM **PAROLE CASUALI DI
          3 LETTERE **
  20 LET A$="ABCDEFGHIJKLMNOPQRS
TUVWXYZ"
  30 PRINT "UN ALTRA LISTA? SI/N
O ";
  40 INPUT R$
  50 PRINT R$
  60 IF R$<>"SI" THEN GOTO 210
  70 REM ** 20 PAROLE **
  80 FOR K=1 TO 20
  90 REM **STRINGA DELLA PAROLA
          VUOTA ALL INIZIO **
 100 LET W$=""
 110 REM **FORMAZIONE DELLA
          PAROLA **
 120 FOR J=1 TO 3
 130 LET X=INT (26*RND+1)
 140 LET W$=W$+A$(X)
 150 NEXT J
 160 REM **STAMPA LA PAROLA**
 170 PRINT W$
 180 REM **TORNA INDIETRO PER LA
          PROSSIMA PAROLA **
 190 NEXT K
 200 GOTO 30
 210 STOP
```

Programma 22

```
RUN
UN ALTRA LISTA? SI/NO SI
GGM
UVZ
DCM
OMV
JFC
XPK
WZG
UQJ
RZT
PFN
WMW
GSU
NGK
JKX
LRF
FPC
KOX
NNY
YXZ
DGC
UN ALTRA LISTA? SI/NO NO
```

Quante volte dovete far girare il programma per ottenere una parola 'propria'?

K Programma 22.

CAPITOLO 7

ELABORAZIONE NUMERICA

7.1 Introduzione

Abbiamo dimostrato che il computer non è soltanto un 'divoratore' di numeri, ma in questa unità daremo uno sguardo più accurato alle sue capacità aritmetiche. Ci concentreremo su un'aritmetica abbastanza semplice, per cui non abbiate timore di trovarvi in difficoltà. Siamo certi che sarete in grado di seguire tutto. Continueremo con il metodo del vedere cosa avviene con determinate operazioni e di ottenere dalla macchina tutte le informazioni su cosa sta facendo. Speriamo che adotterete lo stesso metodo con il vostro ZX81.

7.2 Medie e media aritmetica

'Quanto tempo avete impiegato per leggere ciascuna Unità del corso fino ad ora?' 'Pensiamo circa tre ore in media'. Se vi chiediamo quanto tempo impiegate per andare e venire ogni giorno, per esempio, dal lavoro dovreste rispondere in maniera simile. Gli appassionati di sport usano il termine di media dei gol, gli atlanti sono pieni di immagini sulla caduta media di pioggia nei vari mesi dell'anno. Parliamo di media dei voti di un test o di età media di un gruppo di persone, ecc.

Se desideriamo calcolare la media o media aritmetica dell'età di un particolare gruppo, dovremmo sommare le età di tutti i membri del gruppo e dividere il risultato per il numero totale delle persone del gruppo. Trovare la media aritmetica richiede quindi: sommare, contare, e quindi dividere la somma per il conto.

Esempio 1

Troviamo la media aritmetica della seguente serie di numeri:
6, 7, 2, 5, 4, 4, 9, 8.

Soluzione

La loro somma è = 6+7+2+5+4+4+9+8= 45.
Ci sono 8 numeri.
Per cui la loro media aritmetica è = 45/8 = 5.625

TEST 1

Trovate la media aritmetica della seguente serie di numeri:
8, 4, 2, 6, 1, 7, 6, 1, 4.

Media aritmetica

Esempio 2

Sviluppiamo un algoritmo e scriviamo un programma per trovare la media aritmetica dei seguenti numeri:

56,47,52,65,24,34,59,37,49,66,
38,24,62,76,31,47,66,61,74,45,
66,44,55,67,36,56,54,54,50,43,
18,83,23,79,29,-9999

Soluzione

Esprimeremo l'algoritmo per prima cosa in forma descrittiva.

1. Inizio.
2. Contatore ad 1.
3. Somma a 0.
4. Inserire il prossimo numero.
5. Se il numero è = -9999 allora va alla riga 9 altrimenti continua con la riga 6.
6. Aggiunge il numero alla somma.
7. Aggiunge 1 al contatore.
8. Va alla riga 4.
9. Totale = contatore-1.
10. Calcola la media = somma/totale.
11. Stampa la media.

Figura 1 Algoritmo descrittivo per la media aritmetica

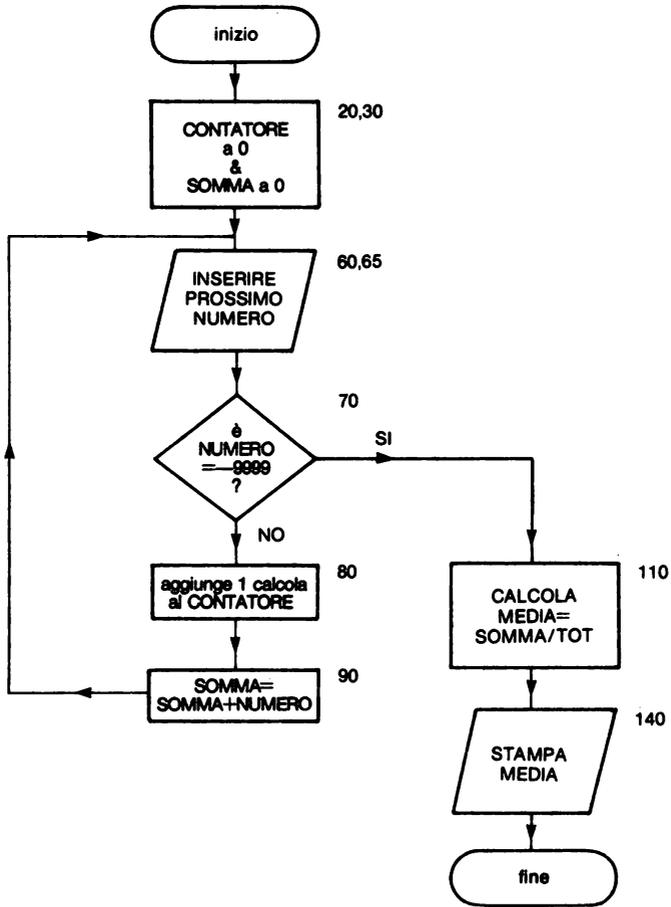



Figura 2 Diagramma di flusso per la media aritmetica

```
  10 REM **MEDIA ARITMETICA**
  20 LET C=0
  30 LET S=0
  40 CLS
  60 PRINT "INSERIRE IL PROSSIMO
 NUMERO,    TERMINARE LA LISTA C
ON -9999"
  65 INPUT N
```

```
 70 IF N=-9999 THEN GOTO 110
 80 LET C=C+1
 90 LET S=S+N
100 GOTO 40
110 LET A=S/C
120 CLS
140 PRINT "MEDIA = ";A
150 STOP
```

Programma 1 Media aritmetica

```
RUN

MEDIA = 50.571428
```

K Programma 1.

Esercizio 1

Scrivete un programma per trovare la lunghezza media delle parole usate nell'Esempio 2 dell'Unità 5. Potete fare questo innestando una routine dentro il Programma 1 dell'Unità 5 il quale trova già la lunghezza delle parole.

Esercizio 2

Scrivete un programma per trovare il punteggio medio in un esperimento simulato di lancio di un dado per 100 volte.

Simulazione

Il punteggio medio teorico quando si lancia un dado per molte volte è:

$$\frac{1+2+3+4+5+6}{6} = \frac{21}{6} = 3.5$$

Tuttavia, nel far girare il programma realizzato nell'Esercizio 2, abbiamo avuto solo una volta 3.5 come risultato, con valori che variavano da 3.24 a 3.76. Ciò come risultato di 3000 (30 x 100) lanci, così potreste chiedere, 'Il generatore di numeri casuali viene influenzato?' (Noi comunque l'abbiamo chiamato 'pseudo' casuale!) Di quanti esperimenti abbiamo bisogno per convincerci che il generatore è, o non è, casuale?

Per esplorare completamente tale quesito dovremmo fare ricorso alla teoria della statistica il che non rientra negli scopi di questo corso, possiamo almeno trovare la media di queste medie. Tutto ciò che dobbiamo fare è di prendere i risultati delle 30 volte che abbiamo fatto girare il programma:
3.56 , 3.47 , 3.52 , 3.65 ,...
ed inserirli nel Programma 1 già descritto. Questo ci da la media aeile medie.

Voi noterete qui in basso che abbiamo inserito solo la parte decimale dei numeri per rendere più semplice l'inserimento dei dati, per es. 56 invece di 3.56. (Possiamo farlo perché la parte intera di tutti i numeri è 3.)

```
56,47,52,65,24,34,59,37,49,66,
38,24,62,76,31,47,66,61,74,45,
66,44,55,67,36,56,54,54,50,43,
-9999

RUN
MEDIA = 50.571428
```

Figura 3

Quindi la media globale di 3000 lanci è tra 3.51 e 3.52 - quindi un po' più rassicurante.

Conpendio sulla simulazione

Simulazione è una parola piuttosto impegnativa per ciò che abbiamo appena fatto. Abbiamo cercato di mettere in evidenza il fatto che possiamo simulare un'attività presente nella realtà senza dover coinvolgere la statistica in maniera profonda. Non possiamo nella realtà lanciare un dado per 3000 volte, ma con un computer possiamo produrre ed elaborare i dati molto rapidamente.

Se abbiamo stimolato la vostra curiosità allora provate il seguente esercizio.

Esercizio 3

Scrivete un programma per trovare il punteggio medio in un esperimento simulato di lancio di 2 dadi per 100 volte.

Quale dovrebbe essere il punteggio medio teorico in questo caso? ed in un esperimento con 3, 4 o più dadi? Ed i risultati del vostro esperimento corrispondono alle vostre attese?

7.3 Oscillazione dei valori

Durante la discussione dei risultati dell'Esercizio 2 nella pagina precedente, abbiamo usato abbastanza naturalmente il concetto di oscillazione dei valori. Si è detto che i valori oscillavano tra 3.24 e 3.76. Il processo ha come conseguenza la ricerca del valore più basso e del valore più alto della serie.

Esempio 3

Sviluppiamo un algoritmo e scriviamo un programma per trovare i valori massimo e minimo dei numeri utilizzati nell'Esempio 2. Aggiungiamo questa routine al programma per trovare la media aritmetica scritto come soluzione dell'Esempio 2. (Se volete, trattate questo Esempio come un esercizio prima di passare a leggere la nostra soluzione.)

Soluzione

Ricorderete che abbiamo fatto qualcosa di simile nell'Unità 4, quando abbiamo trovato il valore più basso in una lista di numeri. Possiamo usare di nuovo tale metodo, per far ciò dovremmo per prima cosa mettere i dati in forma di lista e quindi riordinarli due volte con una routine di interscambio. Ciò richiede molto lavoro, per cui ci rivolgeremo ad un metodo più breve; provare a trovare il numero più basso ed il più alto appena il dato viene inserito.

Noi sappiamo come inserire dei dati, ma cosa dobbiamo fare di tali dati come vengono inseriti?

- Per prima cosa creiamo due locazioni di memoria:
  T per il numero più alto,
  B per il numero più basso.

- Quindi inseriamo il primo numero e lo mettiamo sia in B che in T. (Dopotutto esso è insieme il valore più basso e più alto inserito fino a quel momento
- Poi inseriamo ciascun numero e se è più grande di T, lo mettiamo in T mentre se è più piccolo di B, lo mettiamo in B. Se il numero non è ne più grande di T ne più piccolo di B, passiamo ad inserire il prossimo numero.

Per cui una soluzione descrittiva è:

1. Inizio della routine.
2. Se il contatore e=1 allora scrive il numero in 'T' e in 'B' e va alla riga 6,

altrimenti continua con la riga 3.

3. Se il numero è > di 'T' lo scrive in 'T' e va alla riga 6, altrimenti continua con la riga 4.
4. Se il numero è > = 'B' va alla linea 6, altrimenti va alla riga 5.
5. Scrive il numero in 'B'.
6. Fine della routine.

Figura 4

Ed il diagramma di flusso è:

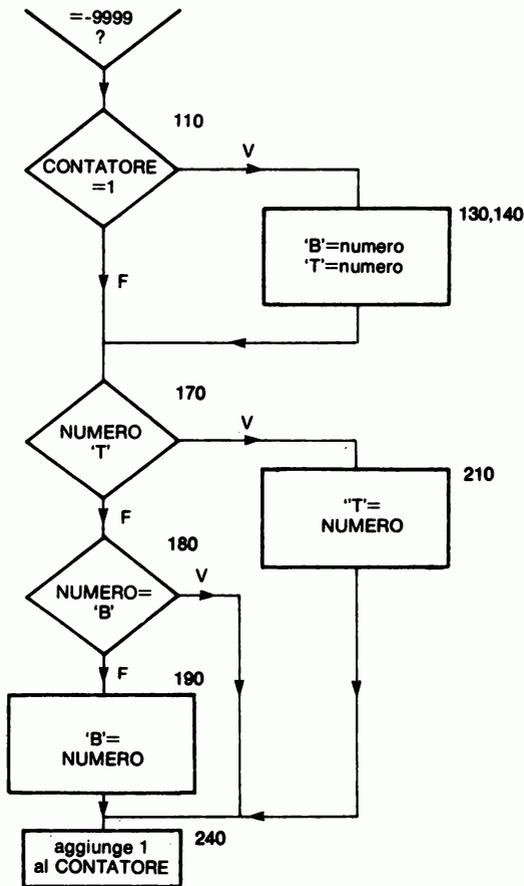



Figura 5

Quale trovate più facile da seguire?

Quando ci sono diversi salti in un programma, l'algoritmo descrittivo può essere piuttosto confuso. La visione bi-dimensionale del diagramma di flusso invece, può essere di maggiore aiuto. È una questione di scelta personale; giudicate voi stessi!

```
 10 REM **MASSIMO E MINIMO**
 40 LET C=1
 60 LET S=0
 70 PRINT "PROSSIMO NUMERO"
 80 INPUT M
 90 PRINT M
100 IF M=-9999  THEN GOTO 300
110 IF C>1 THEN GOTO 170
120 REM *********************
130 LET B=M
140 LET T=M
150 REM *********************
170 IF M>T THEN GOTO 210
180 IF M>=B THEN GOTO 240
190 LET B=M
200 GOTO 240
210 LET T=M
220 REM *********************
240 LET C=C+1
250 LET S=S+M
260 GOTO 70
270 REM *********************
300 PRINT "MAX= ";T,"MIN= ";B
310 REM *********************
320 LET N=C-1
340 PRINT "MEDIA= ";S/N
360 REM *********************
```

il programma passa da questa parte solo all'inserimento del primo numero quando C = 1.

le decisioni vengono prese qui

Programma 2

RUN

```
MAX= 83              MIN= 18
MEDIA= 50.571429
```

In questo esempio è stata omessa la stampa sullo schermo dei numeri inseriti.

K   Programma 2.

Esercizio 4

Scrivete un programma per disegnare la tabella delle frequenze per i dati del Programma 2, usando le categorie:

0-9 , 10-19 , 20-29 ,... 90-100

Suggerimenti

Potreste usare una lista-valori come

S(0), S(10), S(20), ... S(100)

e per ciascun numero che viene letto, verificare se è minore del valore più alto della seconda fascia (10), minore del valore più alto della terza fascia (20), ecc. finché non trovate che

NUMERO < K è vero

quindi incrementate S(K-10); questo è il metodo che abbiamo usato (vedere la risposta).

7.4 Calcolo numerico

Finora in questo corso abbiamo evitato qualsiasi cosa che non fosse della aritmetica abbastanza semplice, e continueremo a fare così. Sarebbe sbagliato non dare un breve sguardo alle capacità aritmetiche del computer. Se ritenete troppo difficile seguire le prossime due-tre pagine, non perderete nessuna funzione vitale di programmazione passando direttamente alla prossima sezione sul funzionamento simulato. Speriamo che proverete a seguirci ugualmente lo ZX81 fa certamente la parte più faticosa del calcolo aritmetico.

Qui c'è un semplice programma che calcola, per i numeri da 1 a 10, il loro quadrato (riga 50), il cubo (riga 60) ed i reciproci (riga 70), e quindi stampa i risultati in forma di tabella.

```
   1 REM **TABELLA DEI QUADRATI,
         CUBI E RECIPROCI DEI
         PRIMI DIECI NUMERI
         NATURALI **
  10 PRINT "N";TAB (5);"N*N";TAB
 (12);"N*N*N";TAB (22);"1/N"
  20 PRINT
  30 FOR I=1 TO 10
  40 LET N=I
  50 LET S=I*I
  60 LET C=I*I*I
  70 LET R=1/I
  80 PRINT N;TAB (5);S;TAB (12);
C;TAB (22);R
  90 NEXT I
 100 STOP
```

Programma 3

```
RUN
N    N*N    N*N*N     1/N

1    1      1         1
2    4      8         0.5
3    9      27        0.33333333
4    16     64        0.25
5    25     125       0.2
6    36     216       0.16666667
7    49     343       0.14285714
8    64     512       0.125
9    81     729       0.11111111
10   100    1000      0.1
```

K Programma 3.

Elevazione alla potenza

Pensiamo che abbiate familiarità con la notazione:

$4x4 = 4^2$ (4 al quadrato o 4 elevato alla 2nda potenza)
$7x7x7 = 7^3$ (7 al cubo o 7 elevato alla 3rza potenza)
$10x10x10x10x10 = 10^5$ (10 elevato alla 5nta potenza)

In BASIC, l'elevazione alla potenza è indicata con ↑ (o semplicemente∧) e sullo ZX81 con **; (tenete presente che ** non sono due normali asterischi ma un simbolo speciale presente sul tasto H).

Qualsiasi numero N elevato alla potenza P viene indicato con N ↑ P dove P viene chiamato esponente.

Allo stesso modo per le potenze negative:

$$\frac{1}{4} \times \frac{1}{4} = \frac{1}{4 \times 4} = 4^{-2}$$

$$\frac{1}{7} \times \frac{1}{7} \times \frac{1}{7} = \frac{1}{7 \times 7 \times 7} = 7^{-3}$$

$$\frac{1}{10} \times \frac{1}{10} \times \frac{1}{10} \times \frac{1}{10} \times \frac{1}{10} \times \frac{1}{10} =$$

$$= \frac{1}{10 \times 10 \times 10 \times 10 \times 10} = 10^{-5}$$

N	P	N ↑ P
4	—2	4 ↑ (—2)
7	—3	7 ↑ (—3)
10	—5	10 ↑ (—5)
10	—5	10 ↑ (—5)

Sono possibili anche le potenze frazionarie (positive e negative) ma non siamo qui interessati ed esse.

Possiamo usare nel Programma 3 l'elevazione alla potenza (**) invece della moltiplicazione (*). Per cui il Programma 3 riscritto con ** diventa:

```
   10 PRINT "N";TAB (5);"N**2";TA
B (12);"N**3";TAB (22);"N**(-1)"
   20 FOR N=1 TO 10
   30 PRINT N;TAB (5);N**2;TAB (1
2);N**3;TAB (22);N**(-1)
   40 NEXT N
   50 STOP
```

Programma 4

Sequenze e loro somma

Calcolare i singoli termini di una sequenza, o la somma dei primi N termini, è un lavoro molto grosso senza un computer. Quanto tempo vi occorrerebbe per per valutare i termini di

$$\frac{1}{1^2}, \frac{1}{2^2}, \frac{1}{3^2}, \ldots \frac{1}{N^2} ?$$

Bene, con il Programma 5 è tutto più facile.

```
30 PRINT "N","N**(-2)"
35 PRINT
50 FOR N=1 TO 10
60 PRINT N,N**(-2)
70 NEXT N
80 STOP
```

Programma 5

```
RUN
N                  N**(-2)

1                  1
2                  0.25
3                  0.11111111
4                  .0625
5                  .04
6                  .027777778
7                  .020408163
8                  .015625
9                  .012345679
10                 .01
```

K Programma 5.

Esercizio 5

Scrivete un programma per scoprire quanti termini della serie

$$1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$$

sono necessari per fare in modo che la loro somma superi 2.4.

Esercizio 6

Modificate il programma dell'Esercizio 5 per scoprire quanti termini sono necessari alla somma

$$1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+.$$

per superare 1.5.

Esercizio 7

Dei numeri interessanti sono i Fattoriali. Fattoriale 4= 4x3x2x1 ed è usualmente scritto così:

Fattoriale N = Nx(N-1)x(N-2)x...x1 = N!

Scrivete un programma per calcolare il Fattoriale di qualsiasi numero positivo intero.

7.5 Funzionamento simulato

Spesso scopriamo che un programma non funziona del tutto, o non è di nostra completa soddisfazione. Se abbiamo a disposizione un computer possiamo sederci davanti alla tastiera per tutto il tempo necessario a rintracciare il difetto, ma quando tale ricerca fallisce possiamo essere costretti a procurarci carta e matita ed a cercare il difetto mentalmente. Esaminare un algoritmo linea per linea con carta e matita viene chiamato funzionamento simulato o prova logica.

Illustreremo una prova logica tramite l'osservazione del Programma 18 che era la soluzione dell'Esercizio 5.

```
 10 REM **SOMMA DI RECIPROCI**
 20 LET S=0
 30 LET N=1
 50 LET S=S+1/N
 60 IF S>2.4 THEN GOTO 100
 70 LET N=N+1
 80 GOTO 50
100 PRINT "SOMMA = ";S
110 PRINT "IL NUMERO DI TERMINI
E ";N
120 STOP
```

```
RUN
SOMMA = 2.45
IL NUMERO DI TERMINI E 6
```

'Tracciare' significa trovare e registrare ciascun passo, il numero di riga di quel passo ed i valori delle variabili dopo che tale riga è stata eseguita. Così, per il Programma 18 abbiamo bisogno di ricordare i valori:

passo N.	N.ro riga	N	S
1			
2			
ecc.			

Abbiamo omesso dalla traccia le linee che non influenzano le variabili, vale a dire:

REM	(riga 10)
GO TO	(riga 80)
PRINT	(riga 100)

A parte ciò, i passi di programma sono i seguenti:

passo N.	N.ro riga	N	S
1	20	0	0
2	30	1	0
3	50	1	1
4	60	1	1
5	70	2	1
6	50	2	1.5
7	60	2	1.5
8	70	3	1.5
9	50	3	1.83
10	60	3	1.83
11	70	4	1.83
12	50	4	2.08
13	60	4	2.08
14	70	5	2.08
15	50	5	2.28
16	60	5	2.28
17	70	6	2.28
18	50	6	2.45

Notate l'effetto di GO TO 50 alla riga 80 (passi 5–6)

GO TO di nuovo (passi 8–9)

GO TO (passi 11–12)

GO TO (passi 14–15)

GO TO (passi 17–18)

Figura 6

Tracciamento

Alcuni interpreti BASIC dispongono del comando TRACE, ma lo ZX81 no. Questa non è una grave lacuna, dato che tale comando spesso fornisce così tante informazioni che è difficile distinguere le informazioni utili da quelle superflue. Una routine di tracciamento progettata con cura spesso lavora meglio. Non mostreremo, quindi, il comando TRACE al lavoro ma come scrivere le vostre routines di tracciamento all'interno di un programma.

TEST 2

Scrivete il funzionamento simulato del seguente programma, iniziando quando la riga 20 è stata appena eseguita e continuando finché la condizione indicata nella riga 60 è vera. Disegnate la tabella nello stesso modo dell'esempio appena descritto.

```
 10 REM **SOMMA DI QUADRATI**
 20 LET S=0
 30 LET N=1
 50 LET S=S+N**2
 60 IF S>50 THEN GOTO 120
 70 LET N=N+1
 80 GOTO 50
100 PRINT "SOMMA = ";S
110 PRINT "IL NUMERO DI TERMINI
E ";N
120 PRINT "SOMMA = ";S
130 PRINT "IL NUMERO DI TERMINI
E ";N
140 STOP
```

Programma 6

7.6 La rappresentazione dei numeri

Finora nel corso abbiamo tenuto in sospeso alcune domande sulla rappresentazione dei numeri nei nostri programmi. Non vogliamo considerare la matematica della rappresentazione dei numeri nei computers in generale, ma abbiamo bisogno di riordinare le nostre idee sui numeri.

In termini generali i computers devono essere in grado di elaborare e memorizzare i seguenti tipi di numeri:

(a) numeri interi positivi e negativi (numeri interi o di conteggio);
(b) frazioni e numeri che sono in parte interi ed in parte frazionari (numeri di misura);
(c) numeri molto grandi e molto piccoli;
(d) il numero zero.

Il principale consiglio che possiamo darvi a questo punto è che se un numero è stato utilizzato in un qualsiasi calcolo all'interno di un programma allora esaminatelo con una certa dose di diffidenza. La ragione di questo suggerimento è che i numeri all'interno del computer vengono memorizzati e manipolati in forma binaria, che è in base 2, piuttosto che in base decimale. Nella familiare notazione decimale ricorderete che un numero del tipo di 1/3 o 1/7 non è in grado di dare un'espressione esatta, per es. 1/3 = 0.33333...; diamo per scontato che aggiungendo al numero tanti 3 quanti ne vogliamo, possiamo ottenere un accettabile grado di accuratezza in qualsiasi particolare problema. Nel sistema binario, allo stesso modo, alcuni numeri non possono essere espressi esattamente, per es. nella forma decimale possiamo dire che 1/10 = 0.1 esattamente, ma quando il numero viene cambiato in forma binaria non può essere rappresentato esattamente.

Molti interpreti BASIC consentono di esprimere i numeri con una precisione di sei cifre decimali. In forma decimale, quindi, il numero 3 e 1/10 può essere rappresentato come 3.10000 con una precisione di sei cifre. Tuttavia, se questo numero è il risultato di alcuni calcoli all'interno del computer, potremmo trovare che il numero risultante è 3.09999 o 3.10001. Quindi, in generale, dovete sempre diffidare delle cifre meno significative di qualsiasi risultato numerico (vale a dire le cifre all'estrema destra del numero).

Il Programma 7 mostra come può capitare questo tipo di imprecisione. Il ciclo FOR ... NEXT aggiunge 4.0 , 4.1 e 4.25 nelle locazioni S, T, U per un migliaio di volte. Ora 4.0 e 4.25 sono rappresentati esattamente in forma binaria, ma 4.1 no. Possiamo vedere che il risultato di questa somma ripetuta è esatto nel caso di 4.0 e 4.25, ma non lo è per 4.1 dove l'errore è di 0.0002 su 4100.00. Ovviamente, dovremmo evitare dove è possibile di scrivere programmi che richiedano tali ripetizioni, ma speriamo che questo programma illustri le possibili imprecisioni.

```
10 REM **DIMOSTRAZIONE DELLA
       PRECISIONE **
20 LET S=0
30 LET T=0
40 LET U=0
50 FOR I=1 TO 1000
```

```
 60 LET S=S+4.0
 70 LET T=T+4.1
 80 LET U=U+4.25
 90 NEXT I
100 PRINT S;TAB (8);T;TAB (21);
U
110 STOP
```

Programma 7

RUN

```
4000      4100.0002     4250
```

K Programma 7.

Numeri grandi e piccoli

I numeri da sei a dieci cifre decimali dei microcomputers non ci consentono di trattare numeri molto grandi o molto piccoli. Così questi numeri sono rappresentati nel BASIC in forma esponenziale.

Numeri piccoli

0.000586321 vuol dire $\frac{586321}{1000000000} = \frac{586321}{10^9}$

che può essere espresso come 586321×10^{-9}. Possiamo anche esprimere 0.000586321

come $\frac{0.586321}{1000} = 0.586321 \times 10^{-3}$

In BASIC questi due ultimi numeri sarebbero scritti come 0.586321E-3.

Allo stesso modo,

$0.0234539 = 2.34539 \times 10^{-2} =$ 2.34539E-2

e

0.00000000959734 = 0.959734E-8.

'E' sta per esponente, e la base di tale esponente è 10. Così E-4 vuol dire muovere il punto decimale di 4 posti a sinistra, ed E+9 vuol dire muovere il punto decimale di 9 posti a destra.

Numeri grandi

12368500 = 1.23685 x 10^7 = 1.23685E+7
935.432 = 0.935432 x 10^3 = 0.935432E+3
959734000000000000000 = 0.959734E+21

La maggior parte degli interpreti BASIC hanno un'estensione numerica che va almeno da E-32 a E+32; comunque dovreste controllare in dettaglio le caratteristiche del particolare sistema che state usando.

```
10 REM **DIMOSTRAZIONE DEI
         NUMERI **
20 PRINT "NUMERO","RAPPRESENTA
ZIONE"
25 PRINT
30 FOR I=-10 TO 10
40 PRINT "10**";I,10**I
50 NEXT I
```

Programma 8

K Programma 8.

7.7 La funzione INT per arrotondare

Per alcuni problemi abbiamo bisogno di arrotondare il risultato di un calcolo al numero intero più vicino.

per es. 6.6 al valore intero più vicino è 7.
7.4 al valore intero più vicino è 7.

Questo è vero specialmente se non siamo a conoscenza delle ultime cifre di precisione di un numero decimale,

per es. 6.99999 o 7.00001 per 7.

La funzione INT (X+0.5) fa questo tipo di arrotondamento per noi, come dimostra il seguente programma.

```
10 REM **INT PER ARROTONDARE**
20 PRINT "X";TAB (10);"INT (X)
";TAB (20);"INT (X+0.5)"
25 PRINT
30 FOR I=-1.4 TO -2.6 STEP (-0
.1)
40 PRINT I;TAB (10);INT I;TAB
(20);INT (I+0.5)
50 NEXT I
60 STOP
```

```
X         INT (X)   INT (X+0.5)

-1.4      -2        -1
-1.5      -2        -1
-1.6      -2        -2
-1.7      -2        -2
-1.8      -2        -2
-1.9      -2        -2
-2        -2        -2
-2.1      -3        -2
-2.2      -3        -2
-2.3      -3        -2
-2.4      -3        -2
-2.5      -3        -2
-2.6      -3        -3
```

Programma 9a

Cambiando la riga 30 con 30 FOR I=1.4 TO 2.6 STEP (.1) otteniamo

```
X         INT (X)   INT (X+0.5)

1.4       1         1
1.5       1         2
1.6       1         2
1.7       1         2
1.8       1         2
1.9       1         2
2         2         2
2.1       2         2
2.2       2         2
2.3       2         2
2.4       2         2
2.5       2         2
2.6       2         3
```

Programma 9b

K Programma 9a e 9b.

TEST 3

Che risultati otterremmo dal programma appena descritto se cambiamo la riga 30 nel modo seguente:

(a) `30 FOR I=.4 TO 2.2 STEP (.2)`
(b) `30 FOR I=.4 TO -.6 STEP (-.2)`

7.8 La funzione ABS

La funzione aritmetica che svolge le idee appena esposte è quella che trova il modulo o valore assoluto di un numero. Ciò suona piuttosto difficile, è invece molto semplice.

> ABS (X) ci da semplicemente il valore positivo di X.
> per es. ABS (23) = 23, ABS (-23) = 23

Il seguente programma illustra meglio la funzione.

```
  10 REM **LA FUNZIONE ABS**
  40 PRINT "X";TAB (7);"Y";TAB (
14);"X+Y";TAB (21);"ABS (X+Y)"
  50 PRINT
  60 FOR I=1 TO 4
  70 INPUT X
  80 INPUT Y
  90 PRINT X;TAB (7);Y;TAB (14);
X+Y;TAB (21);ABS (X+Y)
 100 NEXT I
```

Programma 10

```
RUN
X       Y        X+Y      ABS (X+Y)

5       7        12       12
5       -7       -2       2
-5      7        2        2
-5      -7       -12      12
```

K Programma 10.

TEST 4

Quale sarà il risultato se inseriamo 9, 14, 11, -2, -4, 13, -7, -8?

7.9 Iterazione

Il processo che consiste nel fare un'ipotesi su di un valore, verificarla, correggere l'ipotesi precedente e verificarla di nuovo, ecc. finché non si arriva su un determinato valore, è conosciuto come Iterazione. Un'iterazione comprende:

- la creazione di un'inizio arbitrario
- verificare il grado di precisione di tale punto
- perfezionare tale valore in una sequenza di operazioni ripetute.

Radice quadrata tramite iterazione

Il BASIC può trovare la radice quadrata direttamente, come dimostra questo programma:

```
   20 FOR X=33 TO 63 STEP (10)
   30 PRINT X;TAB (7);X**(0.5);TA
B (20);SQR X
   40 NEXT X
```

Programma 11

(SQR (X) da la radice quadrata di X purché X sia maggiore o uguale a zero.)

```
RUN
33        5.7445627      5.7445627
43        6.5574385      6.5574385
53        7.2801099      7.2801099
63        7.9372539      7.9372539
```

Vi mostreremo anche come trovare una radice quadrata con un metodo iterativo: non perché dobbiate usarlo normalmente, ma perché è un semplice esempio per dimostrare il metodo di iterazione.

Il metodo

Se volete trovare la radice quadrata di un numero N, allora:

- prendete un numero che si avvicini alla radice quadrata, diciamo G
- calcolate il valore di N/G
- adesso calcolate la media tra G ed N/G che dovrebbe essere più indovinato del vostro primo, vale a dire più vicino al valore della radice quadrata sconosciuta del primo tentativo.
- tornate indietro all'inizio usando questo nuovo numero più preciso.

Non proveremo qui questo metodo, (viene descritto nei libri di matematica) ma mostreremo il suo funzionamento in un semplice caso.

Radice quadrata di N=12

Numero approssimato
G = 2

Calcolare N/G = 6

Prendere la media $\frac{G+N/G}{2} = \frac{2+6}{2} = 4$

Così 4 è il nuovo numero approssimato:

Numero approssimatoG = 4

$$N/G = 3$$

$$\frac{G+N/G}{2} = 3.5$$

Quindi 3.5 è il nuovo numero.
In forma di tabella ciò appare come:

G	N/G	$\frac{G+N/G}{2}$
2	12/2=6	(2+6)/2=4
4	12/4=3	(4+3)/2=3.5
3.5	12/3.5=...	....

Figura 7

TEST 5

Per essere sicuri che abbiate afferrato il procedimento, disegnate una tabella simile a quella qui sopra, ma usando 1 come primo numero approssimato.

Fine iterazione

La domanda adesso è: ‘Come facciamo a fermare tale processo?’ Ebbene, noi cerchiamo di fermare il processo quando il quadrato del nostro numero approssimato diventa il più vicino possibile ad N. Ma cosa intendiamo per ’il più vicino possibile?’ Significa che la scelta sta a voi. Voi siete incaricato. Quanto volete che sia precisa la radice quadrata? Se, per esempio, vogliamo trovare la radice quadrata di 12 fino 2 decimali, allora la differenza tra G*G ed N dovrebbe essere

< 0.005

Non abbiamo bisogno di sapere se G*G è più grande o più piccolo di N - conta solo la differenza. Così siamo tornati ad ABS. Se

ABS (N-G*G) < 0.005

allora il processo viene arrestato.

Algoritmo descrittivo per la radice quadrata da iterazione

1. Inizio.
2. Inserire il numero di cui si cerca la radice quadrata.
3. Inserire la precisione richiesta.
4. Inserire un numero approssimato alla radice quadrata.
5. Se il numero rientra nella precisione richiesta allora va alla linea 8, altrimenti continua con la prossima istruzione.
6. Creare un numero più preciso.
7. Ritornare alla linea 5.
8. Stampa il valore di radice quadrata trovato.
9. Fine.

Figura 8

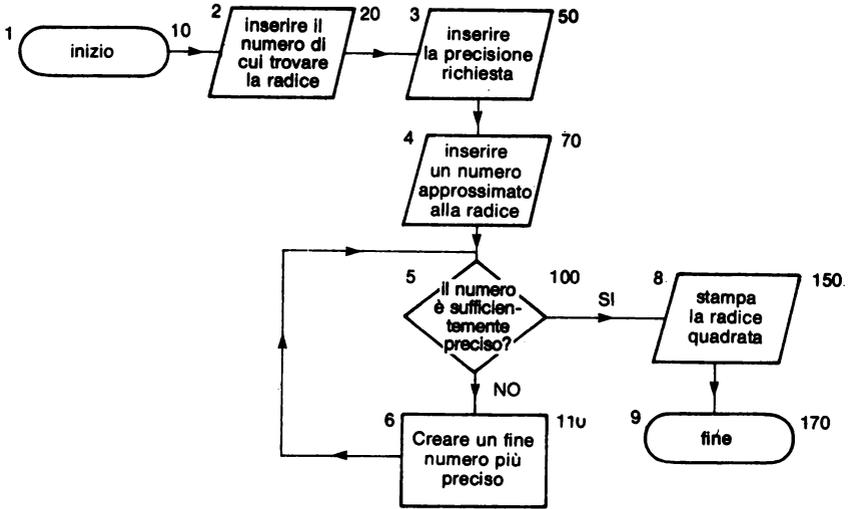



Figura 9 Diagramma di flusso per la radice quadrata da iterazione

```
  10 REM **RADICE QUADRATA DA
            ITEZIONE **
  20 PRINT "NUMERO DI CUI CALC.
LA RADICE"
  30 INPUT N
  40 PRINT N
  50 PRINT "PRECISIONE RICHIESTA
? ";
  60 INPUT A
  65 PRINT A
  70 PRINT "NUMERO APPROSSIMATO?
 ";
  80 INPUT G
  90 PRINT G
 100 IF ABS (N-G*G) <A THEN GOTO
140
 110 LET G=0.5*(G+(N/G))
 120 GOTO 100
 140 PRINT
 150 PRINT "LA RADICE QUADRATA D
I ";N;" E ";G
 160 STOP
```

Programma 12 Radice quadrata da iterazione

RUN

```
NUMERO DI CUI CALC. LA RADICE
12
PRECISIONE RICHIESTA? .005
NUMERO APPROSSIMATO? 2

LA RADICE QUADRATA DI 12 E 3.464
2857
```

RUN

```
NUMERO DI CUI CALC. LA RADICE
12
PRECISIONE RICHIESTA? .00005
NUMERO APPROSSIMATO? 2

LA RADICE QUADRATA DI 12 E 3.464
1016
```

K Programma 12.

Tracciare il processo iterativo

Ora il risultato del Programma 12 non è molto sorprendente. Come abbiamo precisato all'inizio, possiamo trovare la radice di 'N' direttamente con un computer. Ma cercavamo un semplice esempio per mostrare l'iterazione al lavoro così ora daremo uno sguardo più accurato a ciò che è avvenuto. Metteremo una traccia nel Programma 12 nel modo seguente:

```
  95 PRINT
  96 PRINT "VECCHIO";TAB (10);"N
UOVO";TAB (20);"ABS (N-G*G)"

 105 PRINT G;
 115 PRINT TAB (10);G;TAB (20);A
BS (N-G*G)
```

Ad ogni passaggio attraverso il ciclo (vale a dire ad ogni iterazione), le righe 105 e 115 stampano un rapporto su come stanno andando i calcoli.

```
   10 REM **RADICE QUADRATA DA
            ITEZIONE **
   20 PRINT "NUMERO DI CUI CALC.
LA RADICE"
   30 INPUT N
   40 PRINT N
   50 PRINT "PRECISIONE RICHIESTA
? ";
   60 INPUT A
   65 PRINT A
   70 PRINT "NUMERO APPROSSIMATO?
 ";
   80 INPUT G
   90 PRINT G
   95 PRINT
   96 PRINT "VECCHIO";TAB (10);"N
UOVO";TAB (20);"ABS (N-G*G)"
  100 IF ABS (N-G*G)<A THEN GOTO
140
  105 PRINT G;
  110 LET G=0.5*(G+(N/G))
  115 PRINT TAB (10);G;TAB (20);A
BS (N-G*G)
  120 GOTO 100
  140 PRINT
  150 PRINT "LA RADICE QUADRATA D
I ";N;" E ";G
  160 STOP
```

Programma 13 Radice quadrata iterativa con traccia

RUN

```
NUMERO DI CUI CALC. LA RADICE
12
PRECISIONE RICHIESTA? .005
NUMERO APPROSSIMATO? 2

VECCHIO   NUOVO     ABS (N-G*G)
2         4         4
4         3.5       0.24999999
3.5       3.4642857 .0012755059

LA RADICE QUADRATA DI 12 E 3.464
2857
```

solo 3 cicli

RUN

```
NUMERO DI CUI CALC. LA RADICE
12
PRECISIONE RICHIESTA? .00005
NUMERO APPROSSIMATO? 2
```

incrementa la precisione con solo 4 cicli

```
VECCHIO    NUOVO      ABS (N-G*G)
2          4          4
4          3.5        0.24999999
3.5        3.4642857  .0012755059
3.4642857  3.4641016  2.6077032E-8

LA RADICE QUADRATA DI 12 E 3.464
1016
```

K Programma 13.

```
RUN
NUMERO DI CUI CALC. LA RADICE
-67
PRECISIONE RICHIESTA? .005
NUMERO APPROSSIMATO? 8

VECCHIO    NUOVO      ABS (N-G*G)
8          -0.1875    67.035156
-0.1875    178.57292  31955.287
178.57292  89.09886   8005.6068
89.09886   44.173443  2018.2931
44.173443  21.328347  521.8984
21.328347  9.0934941  149.69163
9.0934941  0.86279454 67.744414
0.86279454-38.395923 1541.2469
-38.395923-18.325473 402.82296
-18.325473-7.33468   120.79753
-7.33468   0.90000312 67.810006
0.90000312-36.772092 1419.1867
-36.772092-17.475029 372.37663
-17.475029-6.8204933 113.51913
-6.8204933 1.5014216 69.254267
1.5014216  -21.561476 531.89726
-21.561476-9.2270412 152.13829
-9.2270412-0.98288766
                      67.966068
-0.98288766
           33.5918    1195.409
33.5918    15.798633  316.5968
15.798633  5.7788798  100.39545
5.7788798  -2.9075314 75.453739
```

se inseriamo un numero negativo

la radice non esiste per cui abbiamo dovuto fermare il computer!

Esercizio 8

Cambiate il programma di radice quadrata descritto poco fa in modo da trovare la radice cubica. Se G è il numero approssimato alla radice cubica di N allora 0.5*(G+N/(G*G)) sarà un numero più approssimato.

Obiettivi del capitolo 7

Calcolare (manualmente) una media aritmetica. ☐

Scrivere un programma per calcolare la media aritmetica. ☐

Scrivere un programma per trovare il valore più grande
e più piccolo di una lista di dati. ☐

Usare *, / e ** nei programmi. ☐

Funzionamento simulato di un programma. ☐

Interpretare numeri in notazione E. ☐

Usare INT (X+0.5) per arrotondare. ☐

Usare ABS (X). ☐

Scrivere programmi di routine iterative includendo
delle procedure di arresto. ☐

Inserire in un programma delle righe di tracciamento. ☐

Risposte ai TEST ed agli Esercizi

TEST 1

Somma = 8+4+2+6+1+7+6+1+4 = 39
Ci sono 9 numeri.
Per cui la media aritmetica è 39/9 = 4.333...

Esercizio 1

```
  10 REM **LUNGHEZA MEDIA**
  30 LET S=0
  35 LET C=1
  40 PRINT "PROSSIMA PAROLA"
  50 INPUT W$
  60 PRINT W$
  70 IF W$="ZZZZ" THEN GOTO 140
  80 LET L=LEN (W$)
  90 LET S=S+L
 100 LET C=C+1
 110 GOTO 40
 120 REM ***********************
 140 LET N=C-1
 150 LET A=S/N
 160 PRINT "LA LUNGHEZZA MEDIA D
ELLE PAROLE E DI ";A;" CARATTERI
"
 170 STOP
```

Programma 14

K Programma 14.

Esercizio 2

```
  10 REM **MEDIA DI 100 LANCI
         DI UN DADO **
  40 LET S=0
  60 FOR I=1 TO 100
  70 LET X=INT (6*RND+1)
  80 LET S=S+X
  90 NEXT I
 120 PRINT "PUNTEGGIO MEDIO = ";
S/100
 130 STOP
```

Programma 15

K Programma 15.

Esercizio 3

```
    10 REM **MEDIA DI 100 LANCI
              DI DUE DADI **
    40 LET S=0
    60 FOR I=1 TO 100
    70 LET X=INT (6*RND+1)
    75 LET Y=INT (6*RND+1)
    80 LET S=S+X+Y
    90 NEXT I
   120 PRINT "PUNTEGGIO MEDIO = ";
  S/100
   130 STOP
```

Programma 16

K Programma 16.

Esercizio 4

Questo programma non può girare sullo ZX81 con solo 1K di RAM.

```
  10 PRINT "......ISTOGRAMMA....
.."
  20 DIM S(101)
  40 REM
  45 PRINT
  46 PRINT "DATI...(ESCURSIONE 0
-100)"
  50 INPUT M
  51 IF M=-9999 THEN GOTO 145
  55 PRINT M;" ";
  80 LET K=11
  90 IF M<K THEN GOTO 120
 100 LET K=K+10
 110 GOTO 90
 120 LET S(K-10)=S(K-10)+1
 130 GOTO 50
 145 PRINT
 147 PRINT "TABELLA...."
 148 PRINT
 150 FOR K=1 TO 101 STEP (10)
 160 PRINT K-1;"-";TAB (7);S(K);
TAB (12);
 165 FOR P=1 TO S(K)
 166 PRINT "*";
 167 NEXT P
 170 PRINT
 180 NEXT K
 190 STOP
```

Queste istruzioni (righe 165–167) stampano un istogramma.
Provatele nel vostro programma.

RUN

```
0-      0
10-     1
20-     4
30-     5
40-     7
50-     7
60-     7
70-     3
80-     1
90-     0
100-    0
```

Fate girare il programma usando i dati dell'Esempio 2. Se includete le righe da 165 a 167 avrete anche un istogramma.

K Programma 17.

Esercizio 5

```
 10 REM **SOMMA DI RECIPROCI**
 30 LET S=0
 35 LET N=1
 50 LET S=S+1/N
 60 IF S>2.4 THEN GOTO 110
 70 LET N=N+1
 80 GOTO 50
110 PRINT "SOMMA = ";S
115 PRINT "IL NUMERO DI TERMINI
E ";N
120 STOP
```

Programma 18

RUN

```
SOMMA = 2.45
IL NUMERO DI TERMINI E 6
```

K Programma 18.

Esercizio 6

```
 10 REM **SOMMA DI N**(-2)  **
 30 LET S=0
 35 LET N=1
 50 LET S=S+N**(-2)
 60 IF S>1.5 THEN GOTO 110
 70 LET N=N+1
 80 GOTO 50
110 PRINT "SOMMA = ";S
115 PRINT "IL NUMERO DI TERMINI
E ";N
120 STOP
```

Programma 19

```
RUN
SOMMA = 1.5117971
IL NUMERO DI TERMINI E 7
```

Naturalmente, potete cambiare la riga 60 per scoprire il numero di termini necessari alla somma per superare altri valori, per es.

```
60 IF S>1.6 THAN GO TO 110
RUN
SOMMA = 1.6004969
IL NUMERO DI TERMINI E 22 ———————— per 1.6

60 IF S>1.61 THAN GO TO 110
RUN
SOMMA = 1.611039
IL NUMERO DI TERMINI E 29 ———————— per 1.61

60 IF S>1.62 THAN GO TO 110
RUN
SOMMA = 1.620244
IL NUMERO DI TERMINI E 40 ———————— per 1.62

60 IF S>1.63 THAN GO TO 110
RUN
SOMMA = 1.6301195
IL NUMERO DI TERMINI E 67 ———————— per 1.63
```

K Programma 19.

Esercizio 7

```
 10 REM **FATTORIALI**
 40 PRINT "PER FINIRE, INSERIRE
-9999"
 50 PRINT
 80 PRINT "PROSSIMO FATTORIALE?
";
 85 INPUT N
 86 PRINT N
 90 IF N=-9999 THEN GOTO 200
100 LET F=1
120 FOR I=1 TO N
130 LET F=F*I
140 NEXT I
150 PRINT N,F
160 PRINT
180 GOTO 80
200 STOP
```

Programma 20

```
RUN
PER FINIRE, INSERIRE -9999

PROSSIMO FATTORIALE? 1
1                 1

PROSSIMO FATTORIALE? 3
3                 6

PROSSIMO FATTORIALE? 5
5                 120

PROSSIMO FATTORIALE? 7
7                 5040

PROSSIMO FATTORIALE? 9
9                 362880

PROSSIMO FATTORIALE? 11
11                39916800

PROSSIMO FATTORIALE? -9999
```

K Programma 20.

TEST 2

Passo N.	N.ro riga	N	S
1	20	0	0
2	30	1	0
3	50	1	1
4	70	2	1
5	50	2	5
6	70	3	5
7	50	3	14
8	70	4	14
9	50	4	30
10	70	5	30
11	50	5	55

TEST 3

(a)	X	INT(X)	INT(X+0.5)
	0.4	0	0
	0.6	0	1
	0.8	0	1
	1	0*	1
	1.2	1	1
	1.4	1	1
	1.6	1	2
	1.8	1	2
	2	2	2
	2.2	2	2

*INT I=1 ma si ottiene 0 a causa di imprecisioni di calcolo nel computer.

(b)	X	INT(X)	INT (X+0.5)
0.4	0	0	
0.2	0	0	
0	0	0	
-0.2	-1	0	
-0.4	-1		0
-0.6	-1	-1	

TEST 4

X	Y	X+Y	ABS (X+Y)
9	14	23	23
11	-2	9	9
-4	13	9	9
-7	-8	-15	15

TEST 5

9	N/G	$\frac{G+N/G}{2}$
1	12	6.5
6.5	1.8	4.15
4.15	2.89	3.52
3.52	3.41	3.46 ecc.

Esercizio 8

```
   10 REM **RADICE CUBICA PER
          ITERAZIONE **
   20 PRINT "NUMERO DI CUI TROVAR
E LA RADICE CUBICA?";
   30 INPUT N
   40 PRINT N
   50 PRINT "PRECISIONE RICHIESTA
? ";
   55 INPUT A
   60 PRINT A
   70 LET G=N/2
   80 LET C=1
  100 IF ABS (N-G*G*G)<A THEN GOT
O 140
  110 LET G=0.5*(G+N/(G*G))
  115 LET C=C+1
  120 GOTO 100
  140 PRINT "NUMERO DI CICLI = ";
C
  150 PRINT "LA RADICE CUBICA DI
";N;" E ";G
  160 STOP
```

Riga 70: il computer sceglie N/2 come primo numero approssimato.

```
RUN
NUMERO DI CUI TROVARE LA RADICE
CUBICA?28
PRECISIONE RICHIESTA? .005
NUMERO DI CICLI = 14
LA RADICE CUBICA DI 28 E 3.03640
96

RUN
NUMERO DI CUI TROVARE LA RADICE
CUBICA?10101
PRECISIONE RICHIESTA? .005
NUMERO DI CICLI = 28
LA RADICE CUBICA DI 10101 E 21.6
16634

RUN
NUMERO DI CUI TROVARE LA RADICE
CUBICA?-937
PRECISIONE RICHIESTA? .005
NUMERO DI CICLI = 21
LA RADICE CUBICA DI -937 E -9.78
54198
```

K Programma 21.

CAPITOLO 8

INTRODUZIONE ALL'ELABORAZIONE DATI

8.1 Introduzione

In questa Unità metteremo in evidenza di nuovo come sia importante dare determinati tipi di ordine ai dati. In particolare, analizzeremo in dettaglio un metodo per ordinare dei dati: la procedura di interscambio per riordinare. Riordinati i dati mostreremo quindi come cercare facilmente gli stessi usando la tecnica di ricerca dicotomica. Quindi vedremo come trattare dei dati in forma di tabulato.

Queste attività ci daranno l'occasione di vedere come le subroutines possono aiutarci a compiere molti di quei compiti ripetitivi che capitano in programmi di qualsiasi dimensione.

8.2 Riordino dati

Nell'Unità 4 abbiamo speso un po' di tempo a discutere i metodi per trovare il valore più basso in una lista. Ciò veniva fatto tramite una procedura di interscambio che metteva il valore più basso nella posizione 1 della lista. Dicemmo poi che si poteva ripetere tale procedura con il resto della lista, sistemando il secondo valore più basso nella posizione 2, ecc. Data l'importanza del riordino di dati tramite interscambio, discuteremo ora questo argomento più in dettaglio.

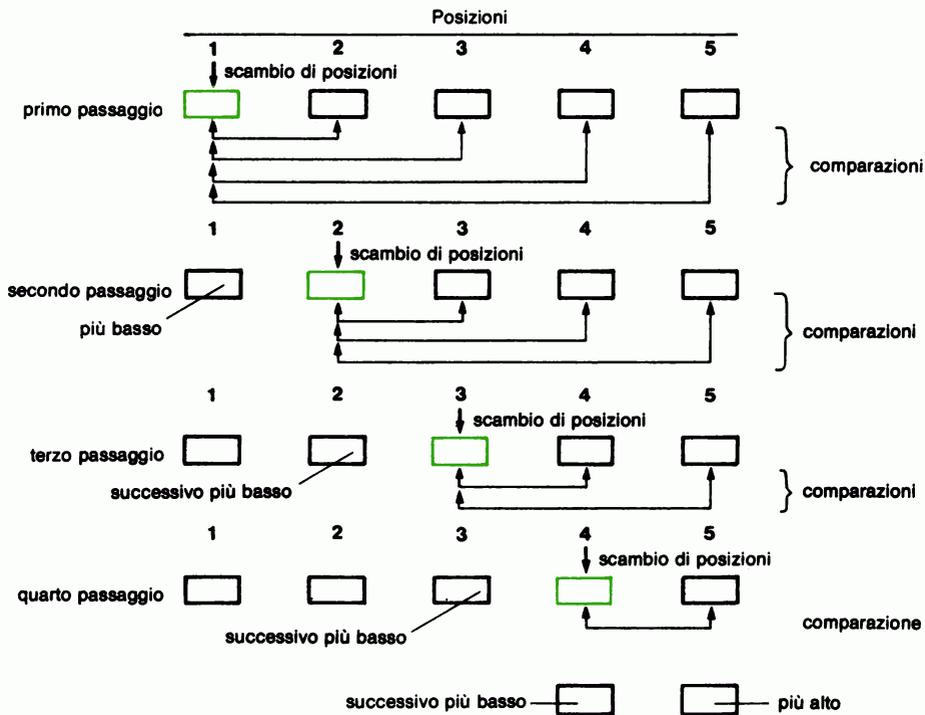



Figura 1 Procedura di riordino per una lista di 5 valori

La Figura 1 illustra la procedura per sistemare i valori all'interno delle locazioni da 1 a 5 con il valore più basso in 1, il successivo valore più basso in 2, e così via.

Primo passaggio. Tutti i valori sono confrontati con il valore nella posizione 1 ed il più basso viene quindi sistemato nella posizione 1.

Secondo passaggio. La posizione 1 ora può essere ignorata e la procedura viene ripetuta con le posizioni da 2 a 5, fino a trovare il successivo valore più basso che viene sistemato nella posizione 2.

Terzo passaggio. Le posizioni 1 e 2 possono essere ignorate poiché esse contengono 'il valore più basso' ed 'il successivo valore più basso'. La procedura viene ripetuta con le posizioni da 3 a 5, per trovare il terzo valore più basso che va alla posizione 3.

Quarto passaggio. Questo viene compiuto solo sui valori 4 e 5 ed il risultante come quarto valore più basso va nella quarta posizione. Il valore rimasto è il più alto e sarà già nella quinta posizione per cui non sono necessari degli altri passi.

Possiamo riassumere la procedura di riordinamento come:

numero ciclo	punto di scambio	sottosequenza rimasta	
		inizio	fine
1	1	2	5
2	2	3	5
3	3	4	5
4	4	5	5

Figura 2a Quattro riordinamenti in una lista di 5 valori

O, più in generale, se vogliamo riordinare una lista di N valori:

numero ciclo	*punto di* scambio	*sottosequenza rimasta*	
		inizio	fine
1	1	2	N
2	2	3	.
3	3	4	.
.	.	.	.
.	.	.	N
.	.	.	.
K	K	K+1	.
.	.	.	.
.	.	.	N
.	.	.	N
N-1	N-1	N	N

Figura 2b (N-1) riordinamenti in una lista di N valori

Poiché ogni passo richiede una serie ripetuta di comparazioni, è un ovvio candidato per un ciclo FOR ... NEXT. Quindi abbiamo bisogno di un ulteriore ciclo per decidere quale dobbiamo utilizzare:

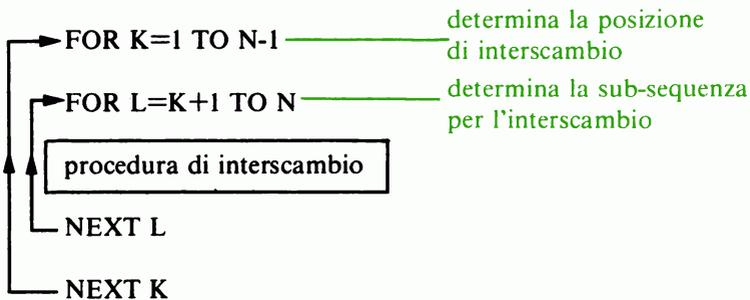



Figura 3 Cicli nidificati del riordinamento con interscambio

TEST 1

Usate il riordinamento con interscambio per mettere in ordine i seguenti valori. Mostrate i numeri memorizzati in ciascuna locazione dopo ogni giro.
6, 1, 4, 0, 2, 3, 7, 8.

La routine è:

il ciclo esterno decide il punto di interscambio

```
200 REM ************************
210 REM **ROUTINE DI RIORDINA-
         MENTO **
220 FOR K=1 TO N-1
230 FOR L=K+1 TO N
240 IF X$(L)>=X$(K)
    THEN GOTO 280
250 LET T$=X$(L)
260 LET X$(L)=X$(K)
270 LET X$(K)=T$
280 NEXT L
290 NEXT K
300 REM **FINE DELLA ROUTINE**
400 REM ************************
```

se il valore nella sub-sequenza è >= al valore nella posizione di interscambio allora l'interscambio non avviene

il 'punto nodale' della routine di riordinamento

il ciclo interno decide la sub-sequenza

Programma 1 Riordinamento con interscambio

Usare il programma di riordinamento

Il programma di riordinamento può essere usato ogni volta che sia necessario. In questo caso particolare per riordinare una lista di nomi in ordine alfabetico.

righe da 50 a 100 leggono i dati
righe da 210 a 300 effettuano il riordinamento
righe da 410 a 460 stampano la lista riordinata

Questo programma non può girare sullo ZX81 con solo 1K di RAM.

```
  10 PRINT ".....ROUTINE RIORDIN
AMENTO....."
  11 PRINT
  15 REM **MASSIMA LUNGHEZZA DA-
            TI = 6 **
  30 DIM X$(20,6)
  50 LET I=1                          ┐
  60 PRINT "PROSSIMO DATO? ";        │
  65 INPUT X$(I)                     ├── legge i dati
  70 PRINT X$(I)                     │
  80 IF X$(I)="ZZZZ  " THEN GOTO     │
 185                                  ┘
  90 LET I=I+1
 100 GOTO 60
 180 REM ************************
 185 REM *LUNGHEZZA DELLA LISTA*
 190 LET N=I-1
 200 REM ************************
 210 REM **ROUTINE DI RIORDINA-      ┐
            MENTO **                  │
 220 FOR K=1 TO N-1                  │
 230 FOR L=K+1 TO N                  │
 240 IF X$(L)>=X$(K) THEN GOTO 2     ├── routine di riordinamento
80                                    │
 250 LET T$=X$(L)                    │
 260 LET X$(L)=X$(K)                 │
 270 LET X$(K)=T$                    │
 280 NEXT L                          │
 290 NEXT K                          │
 300 REM **FINE DELLA ROUTINE**      ┘
 400 REM ************************
 420 PRINT "LISTA FINALE RIORDIN     ┐
ATA"                                  │
 430 FOR P=1 TO N                    ├── stampa il risultato
 440 PRINT X$(P);"  ";               │
 450 NEXT P                          │
 460 PRINT                           ┘
 500 REM ************************
```

Programma 2 Usare la routine di riordinamento

```
RUN
.....ROUTINE RIORDINAMENTO......

PROSSIMO DATO? TANYA
PROSSIMO DATO? SANDRO
PROSSIMO DATO? PIERO
PROSSIMO DATO? LAURA
PROSSIMO DATO? ZZZZ
LISTA FINALE RIORDINATA
LAURA   PIERO   SANDRO TANYA
```

8.3 Subroutines

Arrivati a questo punto vi sarete resi conto che i programmi hanno una struttura costituita dall'assemblaggio di parti più piccole come i paragrafi di un libro. È una pratica consueta spezzare un programma nelle sue parti costituenti, e quindi scrivere e provare ciascuna parte separatamente; alcune operazioni sono spesso ripetute diverse volte all'interno di un programma. La struttura di un programma può essere semplificata e resa più ordinata includendo queste operazioni ripetitive come subroutines (sottoprogrammi).

Illustreremo l'uso delle subroutines dando un tocco finale alla procedura di riordinamento, inserendo altre due serie di righe di traccia nel programma così che si possa vedere cosa avviene in ciascuno dei tre stadi di una routine di riordinamento.

Routine di riordinamento	**Tracciamento per vedere**
1. Inserimento.	La lista come è stata inserita.
2. Riordinamento.	La lista dopo ogni sub-sequenza.
3. Stampa.	La lista finale riordinata.

La Figura 5 mostra la struttura completa ed in che modo possiamo usare una sola subroutine PRINT per tutte e tre le operazioni di stampa.

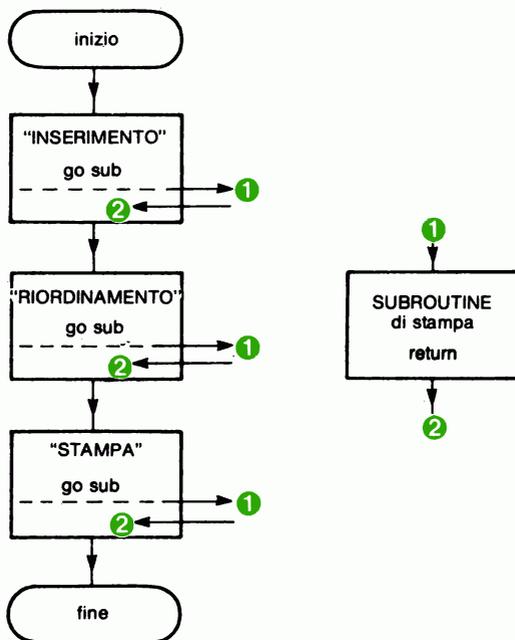



Figura 5 Subroutine di stampa nel programma di riordinamento

GO SUB

Nel BASIC dello ZX81 per saltare alla subroutine usiamo:

GO SUB

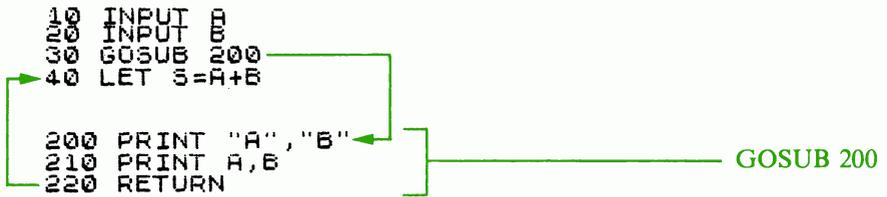
seguito dal numero di riga dell'inizio della subroutine. Ogni subroutine deve terminare con l'istruzione

RETURN

che restituisce il controllo alla riga successiva a quella che contiene l'istruzione GO SUB nella parte principale del programma. Per cui nel seguente segmento di programma la riga 30 trasferisce il controllo alla riga 200, e le righe 200 e 210 vengono eseguite. Quindi la riga 220 ritorna il controllo alla riga 40 in modo che il programma continui per la sua strada normale.

```
10 INPUT A
20 INPUT B
30 GOSUB 200
40 LET S=A+B

200 PRINT "A","B"
210 PRINT A,B
220 RETURN
```

GOSUB 200

TEST 2

Qual'è il valore di B dopo che viene fatto girare il programma: (a) se viene inserito 5 e (b) se viene inserito 3?

```
10 INPUT A
20 IF A<5 THEN GOTO 40
30 GOSUB 80
40 LET B=A*A
50 PRINT B
60 STOP
80 LET A=1/A
90 RETURN
```

Programma 3

Qui di seguito c'è il programma di riordinamento con una routine di stampa (linee 500-550) che viene usata ogni volta che il programma la linea 195, la linea 280 e la linea 430.

```
  10 PRINT "......ROUTINE RIORDIN
AMENTO......"
  11 PRINT
  15 REM **MASSIMA LUNGHEZZA DA-
           TI = 6 **
  30 DIM X$(20,5)
  50 LET I=1
  60 PRINT "PROSSIMO DATO? ";
  65 INPUT X$(I)
  70 PRINT X$(I)
  80 IF X$(I)="ZZZZ " THEN GOTO
185
  90 LET I=I+1
 100 GOTO 60
 180 REM ***********************
 185 REM *LUNGHEZZA DELLA LISTA*
 187 CLS
 190 LET N=I-1
 191 SCROLL
 193 PRINT "LISTA ALL INIZIO"
```

Questo programma non può girare sullo ZX81 con solo 1K di RAM.

Sposta le scritte sullo schermo verso l'alto quando lo schermo è pieno.

```
195 GOSUB 510
196 PRINT
200 REM ***********************
210 REM **ROUTINE DI RIORDINA-
         MENTO **
220 FOR K=1 TO N-1
222 SCROLL
225 PRINT "PASSO N. ";K
230 FOR L=K+1 TO N
240 IF X$(L)>=X$(K) THEN GOTO 2
80
250 LET T$=X$(L)
260 LET X$(L)=X$(K)
270 LET X$(K)=T$
280 GOSUB 510
285 NEXT L
287 PRINT
290 NEXT K
300 REM **FINE DELLA ROUTINE**
400 REM ***********************
405 SCROLL
420 PRINT "LISTA FINALE RIORDIN
ATA"
430 GOSUB 510
450 STOP
500 REM **SUBROUTINE DI STAMPA*
510 SCROLL
515 FOR P=1 TO N
520 PRINT X$(P);"  ";
530 NEXT P
540 SCROLL
550 RETURN
```

Prima traccia (righe 195–196)

Seconda traccia (righe 280–285)

Stampa (righe 430–450)

Subroutine (righe 500–550)

Programma 4 Subroutine di stampa nel programma di riordinamento

```
RUN
LISTA ALL INIZIO
TONI   SUSI   POPI   GINO   BOBO

PASSO N. 1
SUSI   TONI   POPI   GINO   BOBO

POPI   TONI   SUSI   GINO   BOBO

GINO   TONI   SUSI   POPI   BOBO

BOBO   TONI   SUSI   POPI   GINO


PASSO N. 2
BOBO   SUSI   TONI   POPI   GINO

BOBO   POPI   TONI   SUSI   GINO

BOBO   GINO   TONI   SUSI   POPI
```

```
PASSO N. 3
BOBO  GINO  SUSI  TONI  POPI

BOBO  GINO  POPI  TONI  SUSI


PASSO N. 4
BOBO  GINO  POPI  SUSI  TONI


LISTA FINALE RIORDINATA
BOBO  GINO  POPI  SUSI  TONI
```

stampata dal GOSUB alla riga 430

Esempi di subroutines

Lo scopo di una subroutine è di semplificare ed accorciare i programmi lunghi. Per la sua natura, quindi, è difficile ottenere dei programmi brevi e significativi che illustrino le subroutines senza che apparire un po' limitati. Abbiamo bisogno di un programma dove la stessa funzione viene ripetuta in punti diversi del programma.

Esempio 1

Il gioco dei dadi ci fornisce un semplice esempio. Un paio di dadi vengono lanciati due volte ed il punteggio totale di ciascun lancio viene annotato. Se i due punteggi sono uguali il gioco finisce. Se sono differenti, i dadi vengono lanciati di nuovo. Scriviamo ora un programma per simulare il gioco che visualizzi il numero di lanci necessari per ottenere punteggi uguali ed il punteggio stesso.

```
  10 PRINT "......DUE LANCI UGU
ALI......."
  15 PRINT
  30 LET C=1
  35 REM **PRIMO LANCIO **
  40 GOSUB 120
  50 LET S1=S
  55 REM **SECONDO LANCIO **
  60 GOSUB 120
  70 LET S2=S
  80 IF S1=S2 THEN GOTO 100
  90 LET C=C+1
```

La subroutine produce valori di S generalmente differenti per il programma 'principale'.

```
 95 GOTO 35
100 PRINT "PUNTEGGIO UGUALE ";S
1;" IN ";C;" LANCI"
110 STOP
120 REM **SUBROUTINE ROTOLAMEN-
        TO DADI **
130 LET D1=INT (6*RND+1)
135 LET D2=INT (6*RND+1)
140 LET S=D1+D2
150 RETURN
```

Subroutine

Programma 5 Simulazione del gioco dei dadi

RUN

```
.......DUE LANCI UGUALI........

PUNTEGGIO UGUALE 6 IN 2 LANCI

.......DUE LANCI UGUALI........

PUNTEGGIO UGUALE 9 IN 3 LANCI

.......DUE LANCI UGUALI........

PUNTEGGIO UGUALE 7 IN 7 LANCI

.......DUE LANCI UGUALI........

PUNTEGGIO UGUALE 6 IN 4 LANCI

.......DUE LANCI UGUALI........

PUNTEGGIO UGUALE 9 IN 6 LANCI

.......DUE LANCI UGUALI........

PUNTEGGIO UGUALE 7 IN 15 LANCI
```

Esercizio 1

(a) Scrivete un segmento di programma per stampare una linea di 32 lineette "--------" attraverso lo schermo.

(b) Scrivete una riga di programma per stampare un 'sottomarino' oppure <=> in qualsiasi punto dello schermo dove la variabile S, determina la posizione.

(c) Scrivete dunque un programma per stampare su righe successive:
(i) una linea di lineette;
(ii) un sottomarino in un qualsiasi punto;

(iii) un'altra linea di lineette;
con le righe che stampano la (i) e la (iii) in una subroutine.

Esercizio 2

La soluzione dell'Esercizio 1 appare in realtà come una nave in un canale, ma perché non considerarlo un sottomarino? Invece di una nave da guerra in un mare a 2 dimensioni, abbiamo un sottomarino in un canale ad 1 dimensione. Comunque, abbiamo l'immagine per un semplice gioco.

Scrivete un programma per generare un numero casuale tra 1 e 30. Il sottomarino andrà ad occupare le ultime tre posizioni con la sua larghezza (30, 31, 32). Usate il numero casuale per posizionare il sottomarino in una posizione casuale lungo il canale.

Esercizio 3

La ragione del gioco che vogliamo fare con la macchina risulta chiara dall'Esercizio 2. Il computer genera un numero casuale è vi chiede di trovare il sottomarino per tentativi inserendo un numero tra 1 e 29. Se la posizione da voi indicata è esatta, vale a dire tra S e S+2 se S è il numero casuale (ricordate che il sottomarino occupa 3 posti nella linea), allora la macchina stamperà un 'colpito' ed il programma termina. Se non trovate il sottomarino, la macchina stamperà un 'mancato' e vi chiederà di provare di nuovo. Scrivete un programma che faccia tutto ciò.

(Scrivete il programma in modo da avere la possibilità di fermarlo senza dovere per forza trovare il sommergibile.)

8.4 Ricerca dati

Il problema del sottomarino ci può fare da guida nella discussione sulla ricerca dei dati. La sola via metodica per trovare il sottomarino era di cercare nel canale posizione per posizione iniziando da uno dei due estremi. Sarebbe stato più facile se il programma rispondeva con 'troppo alto' o 'troppo basso', secondo i casi, dopo ciascun tentativo. Senza dubbio si può pensare ad una procedura per arrivare sul sommergibile il più rapidamente possibile.

Allo stesso modo, se i dizionari, gli elenchi telefonici, le enciclopedie ed i cataloghi di libri non fossero organizzati in ordine alfabetico, pensate come sarebbe difficile trovare l'informazione desiderata.

Ma se abbiamo speso del tempo per riordinare i nostri dati in ordine numerico o alfabetico, allora abbiamo bisogno di una efficiente tecnica di ricerca per trovare qualsiasi elemento cercato. Se consultiamo un dizionario o un elenco telefonico per cercare un'informazione, non iniziamo guardando sulla prima pagina e quindi leggendo metodicamente il volume pagina per pagina finché non troviamo l'informazione. Facciamo invece un'ipotesi approssimativa, per es. se il nome comincia con 'P' allora proviamo ad aprire il volume appena oltre la metà, ed iniziamo a guardare da quel punto.

Ricerca Dicotomica (bisezione)

Un ipotesi 'approssimativa' è un termine troppo impreciso per un computer. Tuttavia possiamo specificare il punto scelto per tentativo nel modo seguente:

- dividere l'insieme delle voci a metà e domandare 'la voce richiesta è più in alto o più in basso del punto centrale?'
- se è più in basso allora definiamo un nuovo insieme con il punto centrale che ora agisce da limite superiore;
- se è più in alto allora il punto centrale agisce da limite inferiore;
- ogni volta scartiamo metà del precedente insieme e ripetiamo la nostra procedura di dimezzamento o bisezione con il nuovo insieme.

Così, schematicamente, la ricerca con bisezione è:

7 è nella lista 1,2,3,4,5,6,7,8,9,10?

Dimezza la lista. | 6 7 8 9 10 |

7 = valore centrale? No.
7 < valore centrale? Si.

Dimezza la lista. | 6 7 8 |

7 = valore centrale? Si.
7 è nella lista.

Questo schema illustra il principio della ricerca con bisezione. Dobbiamo però distinguere tra il valore degli elementi di una lista e gli indici degli elementi stessi.

Esempio 2

Una lista ordinata contiene le voci A, F, I, M, P, T, U, Z. Usiamo la procedura di ricerca con bisezione per trovare se P è o no nella lista.

Chiamiamo P, 'Quesito' - il valore di cui vogliamo interessarci.

Figura 6 Ricerca con bisezione

Esercizio 4

Utilizzate la procedura di ricerca con bisezione sulla lista dell'Esempio 2 questa volta per cercare la lettera I.

Nell'Esempio 2 abbiamo eseguito solo 3 comparazioni per trovare la voce P, ma può sembrare che l'intera procedura abbia pochi vantaggi su una semplice ricerca attraverso la lista. In realtà l'efficacia del metodo non è evidente con le liste brevi. Dimostreremo più avanti la sua potenza nel cercare in liste lunghe, prima però chiariamo alcuni punti.

Alcuni problemi della ricerca con bisezione
(a) Come fermarsi

Esempio 3

Utilizzate la stessa procedura di prima, per cercare la lettera 'Q'.

Il metodo dovrebbe procedere esattamente come prima fino alla terza comparazione, così noi riprendiamo da quel punto.

Quesito = Q

Indice		4	5	6
Voce		M	P	T
Inizio- Basso		Indice (4)	:	Alto (6)
Indice-Medio, Int	$\frac{(4+6)}{2} = 5$		: Medio (4)	

Comparazione

> Quesito = Voce (5)? no!
> Quesito < Voce (5)? no!
> Indice (5) = nuovo Basso

Indice		4	5	6
Voce			P	T
Inizio- Basso			Indice (5)	Alto (6)
Indice-Medio, Int	$\frac{(5+6)}{2} = 5$		Medio (5)	

Comparazione	Quesito = Voce (5)? no! Quesito < .... non abbiamo già fatto prima questo confronto?

Così non sembriamo in grado di fermarci. Q non c'è, ma continuiamo a cercarlo tra P e T. Abbiamo già incontrato il problema di fermare il processo nell'ultimo esempio. Se gli indici Basso ed Alto si sono avvicinati tanto da essere in posizioni adiacenti, e Quesito non è stato trovato, allora Quesito non fa parte della lista. Questa è la fine e l'uscita dalla ricerca. Così la fine si ha quando Quesito è stato trovato, o quando Basso ed Alto occupano degli indici adiacenti (Alto - Basso = 1).

(b) Come iniziare

Iniziare il processo appare abbastanza semplice. Basta fissare l'Indice(1)=Basso e l'Indice(N)=Alto. Tuttavia potrebbero presentarsi dei problemi se Voce(1) e Voce(N) non fossero il valore più basso e più alto possibile.
Per es. considerate la seguente lista che non include lettere prima di C o dopo S:

1	2	3	4	5
C	F	G	P	S
Basso			Alto	

Se Quesito fosse A o B o più alto di S, il processo non funzionerebbe. La soluzione più facile è di assicurarsi che le voci alle estremità della lista abbiano già i valori estremi, per es. in una lista di nomi usate Voce(1)=AAAA e Voce(N)=ZZZZ.

Ora disegneremo l'algoritmo in forma di diagramma di flusso.

GLOSSARIO
Tutti i termini sono definiti nel testo precedente.

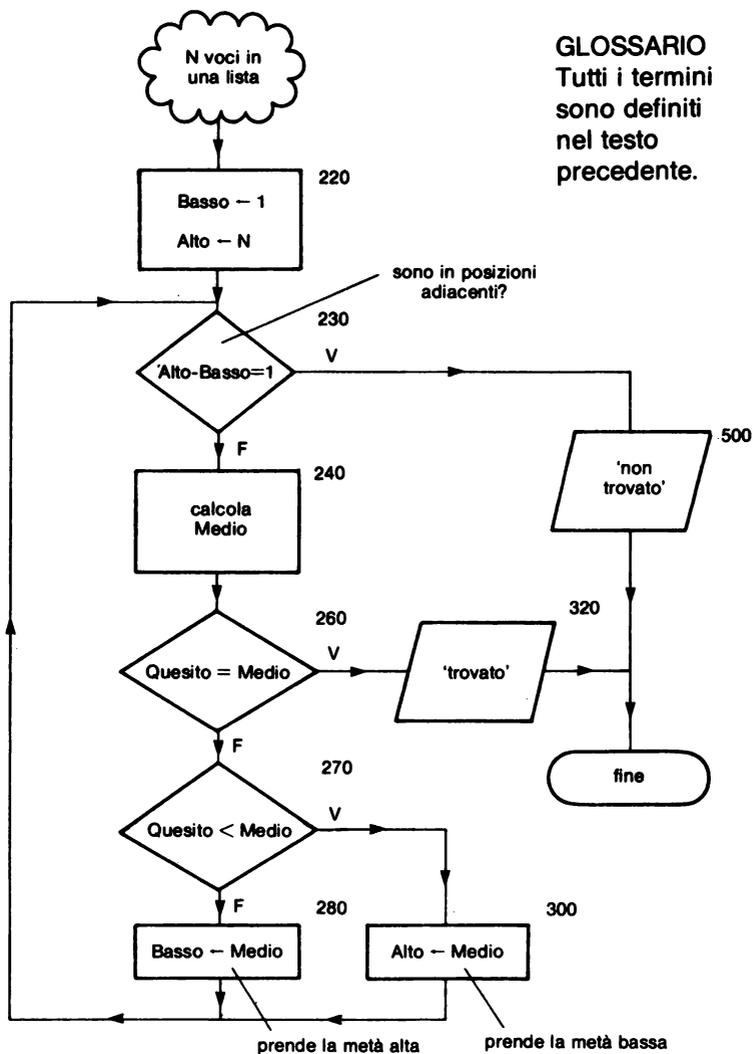



Figura 7 Diagramma di flusso per la ricerca con bisezione

Tutto ciò che dobbiamo fare adesso è di scrivere il programma:

Questo programma non può girare sullo ZX81 con solo 1K di RAM.

```
  10 REM *RICERCA CON BISEZIONE*
  20 DIM N$(13,7)
  25 DIM Q$(7)
  30 DIM T$(13,4)
  40 LET N$(1)="AAAA"
  41 LET N$(2)="BRUNO"
  42 LET N$(3)="CARLO"
  43 LET N$(4)="DARIO"
  44 LET N$(5)="ENZO"
  45 LET N$(6)="LAURA"
  46 LET N$(7)="MARIO"
  47 LET N$(8)="PAOLA"
  48 LET N$(9)="SANDRO"
  49 LET N$(10)="VANNA"
  50 LET N$(11)="VIOLA"
  51 LET N$(12)="WANDA"
  52 LET N$(13)="ZZZZ"
  60 LET T$(1)="0000"
  61 LET T$(2)="1234"
  62 LET T$(3)="9832"
  63 LET T$(4)="1980"
  64 LET T$(5)="7294"
  65 LET T$(6)="5821"
  66 LET T$(7)="8632"
  67 LET T$(8)="7832"
  68 LET T$(9)="1383"
  69 LET T$(10)="1147"
  70 LET T$(11)="5529"
  71 LET T$(12)="9936"
  72 LET T$(13)="9999"
  90 REM
 100 LET N=13
 105 REM **QUESTA VOLTA ABBIAMO
          USATO ZZZZ **
 150 PRINT "NOME CERCATO? ";
 160 INPUT Q$
 170 PRINT Q$
 200 REM **INIZIO DELLA RICRCA**
 205 PRINT " B";TAB (5);" A";TAB
 (10);" M";TAB (15);"N$(M)"
 220 LET B=1
 230 LET A=N
 235 IF A-B=1 THEN GOTO 500
 240 LET M=INT ((B+A)/2)
 250 PRINT B;TAB (5);A;TAB (10);
M;TAB (15);N$(M)
 260 IF Q$=N$(M) THEN GOTO 320
 270 IF Q$<N$(M) THEN GOTO 300
 280 LET B=M
 290 GOTO 235
 300 LET A=M
 310 GOTO 235
```

```
320 REM **FINE DELLA RICERCA**
330 PRINT "IL TELEFONO DI ";Q$;
"E ";T$(M)
350 GOTO 150
500 PRINT Q$;" NON E NELLA LIST
A"
550 GOTO 150
```

Programma 6 Ricerca con bisezione

```
RUN
NOME CERCATO? MARIO
 B     A     M     N$(M)
1     13    7      MARIO
IL TELEFONO DI MARIO  E 8632

NOME CERCATO? VERA
 B     A     M     N$(M)
1     13    7      MARIO
7     13    10     VANNA
10    13    11     VIOLA
VERA     NON E NELLA LISTA
```

8.5 Tabelle

Quando vogliamo memorizzare una certa quantità di informazioni abbiamo a nostra disposizione vari metodi. Uno è quello di usare delle liste (vedere Unità 4), che sono a volte chiamate vettori mono-dimensionali. Un secondo metodo è quello di usare delle tabelle o matrici (dette anche vettori bi-dimensionali).

Supponiamo che vogliate memorizzare i seguenti dati:

	primo quarto	*secondo quarto*	*terzo quarto*	*quarto quarto*
Auto vendute	20	70	80	40
Servizi	10	14	18	11
Benzina	30	45	50	30

Figura 8 Reddito di una stazione di servizio (cifre in Milioni di Lire)

Potete mettere questi dati in una lista ma sarebbe difficile da usare. I primi quattro valori dovrebbero essere le entrate per auto vendute, i successivi quattro le entrate per il servizio di assistenza, ecc. In alternativa potreste avere tre liste: una per le auto vendute, una per i servizi ed una per la benzina. Tutto ciò non occorre perché lo ZX81 vi consente di avere delle tabelle bi-dimensionali il cui nome può essere una qualsiasi delle 26 lettere dell'alfabeto, per es.

T(,)

Confronto tra liste e tabelle

Le liste richiedono solo un indice per segnare una posizione nella lista. Le tabelle ne richiedono due, che sono usualmente chiamati subscritti e non indici.

Vettore (Lista)

V(1) , V(2) , V(3) ... V(N)

indice di questo elemento = 3

Matrice (Tabella)

- Una tabella deve contenere o tutte variabili stringa o tutte variabili numeriche. (I numeri possono naturalmente essere memorizzati come stringhe, ed il loro valore trovato tramite la funzione VAL.)

- Usiamo una delle 26 lettere per descrivere la tabella nel suo intero, per es. tabella A, tabella B$ tabella M$.

	col.1	col.2	col.3	col.4
riga 1	r1c1	r1c2	r1c3	
riga 2	r2c1	r2c2	r2c3	
riga 3	r3c1	r3c2	r3c3	
ecc.	....	....	....	....

Figura 9 Righe e colonne di una tabella

Per i dati della stazione di servizio, T richiede 3 righe e 4 colonne in modo da contenere 12 elementi:

T(1,1)	T(1,2)	T(1,3)	T(1,4)
T(2,1)	T(2,2)	T(2,3)	T(2,4)
T(3,1)	T(3,2)	T(3,3)	T(3,4)

Così

T(2,1) = 10
T(3,3) = 50 ecc.

Questa è molto simile alle tabelle che abbiamo incontrato in precedenza. In questo caso diciamo che un File consiste in una serie di Records ognuno dei quali è formato da Campi. In forma di tabella ciò apparirebbe come:

	Campo 1	Campo 2
	Nome	Telefono
Record 1	BRUNO	1234
Record 2	CLAUDIA	9823
Record 3	DARIO	1850
Record 4	ENZO	7294

O, più in generale:

	Campo-1	Campo-2	Campo-3	ecc.
Record-1	R1F1	R1F2	R1F3	
Record-2	R2F1	R2F2	R2F3	
Record-3	R3F1	R3F2	R3F3	
ecc.	....	....	....	....

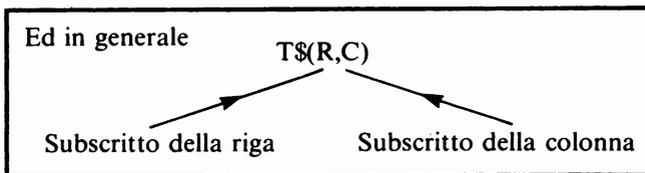
Se la tabella dei numeri telefonici viene chiamata T$, allora i singoli elementi saranno etichettati:

	Campo 1	Campo 2
	Nome	Telefono
Record-1	T$(1,1)= BRUNO	T$(1,2)=1234
Record-2	T$(2,1)= CLAUDIA	T$(2,2)=9823
Record-3	T$(3,1)= DARIO	T$(3,2)=1850
Record-4	T$(4,1)= ENZO	T$(4,2)=7294

- L'intera tabella viene chiamata tabella T$
- Ogni elemento nella tabella viene descritto da due subscritti. Quindi 1850 (riga 3, colonna 2) è

  T$(3,2)
- Il 3 ed il 2 descrivono la posizione dell'elemento T$ (3,2), non il suo valore, che è 1850.

Così diciamo

T$(3,2) = 1850

Esempio 4

Questa tabella N$ ha 9 valori come visibile. Quali sono i nomi delle loro variabili?

R \ C	1	2	3
1	BRUNO	CARLO	DARIO
N$(,) 2	ENZO	GINA	LAURA
3	MARIO	PAOLA	SANDRO

Soluzione

BRUNO=N$(1,1) CARLO=N$(1,2) DARIO=N$(1,3)

ENZO=N$(2,1) GINA=N$(2,2) LAURA=N$(2,3)

MARIO=N$(3,1) PAOLA=N$(3,2) SANDRO=N$(3,3)

TEST 3

Nella seguente tabella A identificate le variabili ed i loro valori come nell'Esempio 4.

ANDREA	BRUNO	CLAUDIA
DAVIDE	ENZO	FRANCO
GIULIA	LAURA	MARIO
NINO	PAOLA	ROSA
SANDRO	TINO	VANNA

DIM per matrici

Matrici numeriche Se volete una matrice numerica a due dimensioni dovete indicare al computer di che dimensioni dovrà essere usando l'istruzione DIM, per es. per memorizzare la tabella

inserite

DIM A(3,4)
colonne
righe

nel programma prima di usare la tabella A( ).

Matrici stringa. Lo stesso si applica alle matrici stringa a due dimensioni stabilendo come con le liste di stringhe la lunghezza massima. Così per memorizzare la tabella

dovrete inserire

DIM A$(3,4,10)
lunghezza massima stringa
colonne
righe

Tabelle e cicli nidificati

Se i cicli FOR...NEXT e le liste sembrano fatti l'una per l'altra, così ancora di più i cicli FOR...NEXT nidificati e le tabelle appaiono complementari.

Per esempio, supponete di voler inserire:

ANDREA, BRUNO, CLAUDIA, DAVIDE, ENZO, FRANCO, GIULIA, LAURA, MARIO, NINO, ORESTE, PAOLA, ROSA, SANDRO, TINO, VANNA

in una tabella, N$, con 4 righe e 4 colonne. (Abbiamo bisogno di una matrice stringa perché dobbiamo memorizzare dei dati alfabetici.) Questo può essere fatto con una istruzione INPUT in due cicli nidificati:

```
 60 FOR I=1 TO 4
 70 FOR J=1 TO 4
 80 PRINT "PROSSIMO DATO? ";
 81 INPUT N$(I,J)
 82 PRINT N$(I,J)
 83 PRINT TAB (30);4*(I-1)+J
 90 NEXT J
100 NEXT I
```

Questo procedimento è riportato al completo nelle righe da 10 a 100 del Programma 7.

Il programma memorizza correttamente i valori in una tabella, ma non possiamo vedere il risultato di ciò finché non viene stampato. La seconda metà del programma stampa la tabella dei valori in una colonna insieme ad I e J in modo che possiate vedere chiaramente come sono stati usati.

```
  10 PRINT ".INSERIMENTO E STAMP
A TABELLA.."
  15 PRINT
  20 PRINT "TABELLA UTILIZZATA P
ER 16 DATI."
  30 DIM N$(20,5,7)
  40 REM **ROUTINE INSERIMENTO
         DATI **
  60 FOR I=1 TO 4
  70 FOR J=1 TO 4
  80 PRINT "PROSSIMO DATO?
  81 INPUT N$(I,J)
  82 PRINT N$(I,J)
  83 PRINT TAB (30);4*(I-1)+J
  90 NEXT J
```

Dimensionamento matrice bidimensionale

```
 100 NEXT I
 110 REM **ROUTINE DI STAMPA**
 111 PRINT
 112 PRINT ".......TABELLA COMPL
ETA......."
 113 PAUSE 100
 114 POKE 16437,255
 115 CLS
 116 PRINT "I";TAB (10);"J";TAB
(20);"N$(I,J)"
 130 FOR I=1 TO 4
 140 FOR J=1 TO 4
 150 PRINT I;TAB (10);J;TAB (20)
;N$(I,J)
 160 NEXT J
 180 NEXT I
 200 STOP
```

Programma 7 Dati in una Matrice 4 x 4

istruzione DIM per matrice a due dimensioni

routine inserimento dati

RUN

```
I         J         N$(I,J)
1         1         ANDREA
1         2         BRUNO
1         3         CLAUDIA
1         4         DAVIDE
2         1         ENZO
2         2         FRANCO
2         3         GIULIA
2         4         LAURA
3         1         MARIO
3         2         NINO
3         3         ORESTE
3         4         PAOLA
4         1         ROSA
4         2         SANDRO
4         3         TINO
4         4         VANNA
```

TEST 4

Al Programma 7 vengono fatte le seguenti modifiche. Scrivete come apparirà la stampa della tabella.

```
 60 FOR I=1 TO 3
 70 FOR J=1 TO 5
130 FOR I=1 TO 3
140 FOR J=1 TO 5
```

Stampa Tabella

La stampa del Programma 7 non è molto soddisfacente poiché vorremmo vedere la tabella in forma più ordinata. Per far questo cancellate le righe dopo 115 ed inserite una nuova routine di stampa:

Programma 8

la prima colonna inizia alla posizione 1-1=0

colonne larghe 10 caratteri

La stampa quindi è:

```
ANDREA    BRUNO     CLAUDIA
ENZO      FRANCO    GIULIA
MARIO     NINO      ORESTE
ROSA      SANDRO    TINO
```

Obiettivi del capitolo 8

Ora che avete completato questa unità, verificate di essere in grado di:

Usare il riordinamento con interscambio (manualmente) su una serie di dati ☐

Scrivere due cicli di programma nidificati per effettuare il riordinamento con interscambio ☐

Seguire GO SUB nei programmi ☐

Scrivere GO SUB all'interno dei programmi ☐

Usare la ricerca con bisezione (manualmente) su una serie di dati ☐

Scrivere un programma per la ricerca con bisezione ☐

Mettere dei dati in una matrice a due dimensioni ☐

Scrivere un programma per inserire dei dati in una matrice a due dimensioni ☐

Scrivere un programma per stampare i dati di una matrice a due dimensioni ☐

Risposte ai TEST ed agli Esercizi

TEST 1

```
0  6  4  1  2  3  7  8
0  1  6  4  2  3  7  8
0  1  2  6  4  3  7  8
0  1  2  3  6  4  7  8
0  1  2  3  4  6  7  8
0  1  2  3  4  6  7  8
0  1  2  3  4  6  7  8
```

LISTA FINALE RIORDINATA

```
0  1  2  3  4  6  7  8
```

TEST 2

(a) B = 1/25
(b) B = 9 (in questo caso GO SUB non viene usato.)

Esercizio 1

(a)
```
10 FOR I=1 TO 32
20 PRINT "-";
30 NEXT I
40 PRINT
```

Programma 9

(b)
```
30 PRINT TAB (S);"<=>"
```

(c)
```
 15 INPUT S
 16 PRINT S
 20 GOSUB 110
 30 PRINT TAB (S);"<=>"
 40 GOSUB 110
 50 STOP
110 FOR I=1 TO 32
120 PRINT "-";
130 NEXT I
140 PRINT
150 RETURN
```

Programma 10

Il valore che diamo ad S determina la posizione del sottomarino lungo il canale, ed otteniamo un'immagine del tipo:

```
9
--------------------------------
         <=>
--------------------------------
```

Esercizio 2

```
10 REM **SOTTOMARINO**
50 LET S=INT (29*RND+1)
60 GOSUB 510
70 PRINT TAB (S);"<=>"
80 GOSUB 510
90 STOP
```

```
500 REM **SUBROUTINE DI STAMPA*
510 FOR I=1 TO 32
520 PRINT "-";
530 NEXT I
540 PRINT
550 RETURN
```

Programma 11

```
RUN
--------------------------------
      <=>
--------------------------------
```

```
RUN
--------------------------------
                     <=>
--------------------------------
```

```
RUN
--------------------------------
               <=>
--------------------------------
```

Esercizio 3

```
 10 REM **SOTTOMARINO**
 30 REM **STAMPA LA SFIDA**
 40 GOSUB 300                              → 1
 50 PRINT "PER TROVARMI OCCORRE
UN NUMERO  DA 1 A 29 "
 60 GOSUB 300                              → 2
 70 REM **POSIZIONE CASUALE DEL
          SOTTOMARINO **
 80 LET S=INT (29*RND+1)
 90 PRINT
100 PRINT "PROVATE UN ALTRO NUM
ERO? ";
105 INPUT X
106 PRINT X
110 IF X<S THEN GOTO 190
120 IF X>S+2 THEN GOTO 190
130 REM **COLPO CENTRATO**
140 GOSUB 300                              → 3
150 PRINT "COLPITO"
160 GOSUB 300                              → 4
```

```
170 GOTO 320
180 REM **COLPO A VUOTO**
190 GOSUB 300                          → 5
200 PRINT "MANCATO"
210 GOSUB 300                          → 6
220 PRINT "VOLETE CONTINUARE A
GIOCARE S/N"
222 INPUT R$
225 PRINT R$
230 IF R$="S" THEN GOTO 90
240 PRINT "SBAGLIATO, IO ERO QU
I"
250 GOSUB 300                          → 7
260 PRINT TAB (5);"<=>"
270 GOSUB 300                          → 8
280 GOTO 320
290 REM **SUBROUTINE DI STAMPA*
300 FOR I=1 TO 32  ┐
305 PRINT "-";     ┘── la subroutine stampa solo una linea di trattini
310 NEXT I
311 PRINT
315 RETURN
320 STOP
```

Programma 12

RUN

```
--------------------------------
PER TROVARMI OCCORRE UN NUMERO
DA 1 A 29
--------------------------------

PROVATE UN ALTRO NUMERO? 5
--------------------------------
MANCATO
--------------------------------
VOLETE CONTINUARE A GIOCARE S/N
S

PROVATE UN ALTRO NUMERO? 18
--------------------------------
MANCATO
--------------------------------
VOLETE CONTINUARE A GIOCARE S/N
S

PROVATE UN ALTRO NUMERO? 26

--------------------------------
MANCATO
--------------------------------
VOLETE CONTINUARE A GIOCARE S/N
N
SBAGLIATO, IO ERO QUI
--------------------------------
     <=>
--------------------------------
```

Esercizio 4

1	2	3	4	5	6	7	8
A	F	I	M	P	T	U	Z

(4)

Quesito = Voce (4)? No.
Quesito < Voce (4)? Si.
Indice (4) = nuovo Alto.

A	F	I	M
1	2	3	4

$$\text{Indice-Medio} = \text{Int}\ \frac{(1+4)}{2} = 2$$

Quesito = Voce (2)? No.
Quesito < Voce (2)? No.
Indice (2) = nuovo Basso.

2	3	4
F	I	M

Quesito = Voce (3)? Si.
Quindi I è nella lista.

TEST 3

A$(1,1)=ANDREA	A$(1,2)=BRUNO	A$(1,3)=CLAUDIA
A$(2,1)=DAVIDE	A$(2,2)=ENZO	A$(2,3)=FRANCO
A$(3,1)=GIULIA	A$(3,2)=LAURA	A$(3,3)=MARIO
A$(4,1)=NINO	A$(4,2)=PAOLA	A$(4,3)=ROSA
A$(5,1)=SANDRO	A$(5,2)=TINO	A$(5,3)=VANNA

TEST 4

RUN

I	J	N$(I,J)
1	1	ANDREA
1	2	BRUNO
1	3	CLAUDIA
1	4	DAVIDE
1	5	ENZO
2	1	FRANCO
2	2	GIULIA
2	3	LAURA
2	4	MARIO
2	5	NINO
3	1	ORESTE
3	2	PAOLA
3	3	ROSA
3	4	SANDRO
3	5	TINO

(Notate che VANNA non è stata inserita nella tabella. Una tabella 5 x 3 può leggere solo i primi 15 elementi.)

Indice Analitico

Il personal computer è la macchina degli anni 80; il BASIC è il suo linguaggio, e imparare il BASIC può aiutare a capire la rivoluzione informatica.
Questo semplice corso di autoistruzione insegna a programmare, e un programma ha sempre bisogno di due ingredienti, un linguaggio e una struttura, dunque questo libro non insegna solo il BASIC, ma anche come si organizza correttamente un buon programma.
Insegna a organizzare un archivio di informazioni e a tenerlo in ordine, a stampare lettere e indirizzi, a progettare un gioco, e altro ancora, presentando e descrivendo le applicazioni più utili e diffuse del calcolatore.

L. 31.000 **Cod. 547B** **ISBN 88-7056-215-8**

164
Il BASIC in 30 ore per ZX81
Clive Prigmore
GRUPPO
EDITORIALE
JACKSON